Introduction to Artificial Intelligence

丁尧相
浙江大学

Fall & Winter 2022
Week 11

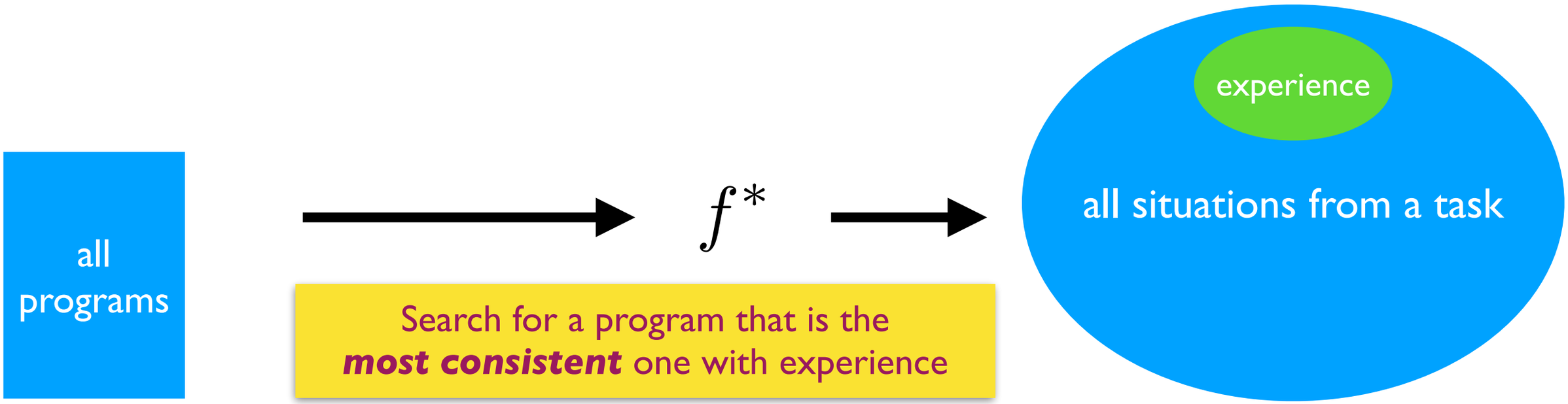
Announcements

- We will release Problem set 3.1-3.2 after this lecture.
- We will release Lab project 2 (knowledge inference) this week.
- The due date of Lab project 1 will be on Dec 2.

Machine Learning: II

- Support vector machine (SVM)
  - Linear separable case
  - Linear inseparable case
- AdaBoost
- Take-home messages

Machine Learning



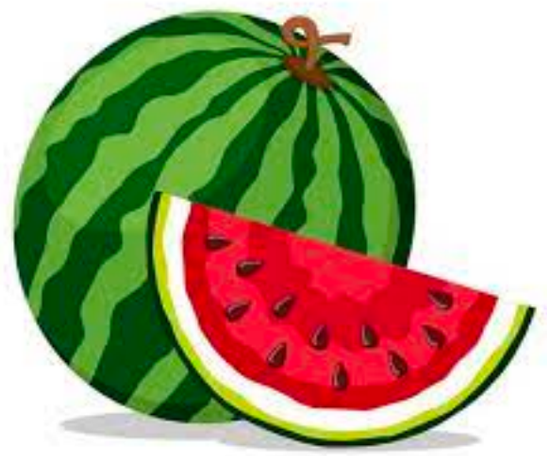
"Machine learning = task + data + objective + algorithm".

— Tom Mitchell

Example: Watermelon Classification

- Build a program to detect whether a watermelon is good or not.
- The program: $f : X \to Y$
  - X : features of watermelon, e.g. color, root type, touch sound.
  - Y : binary variable, indicating good or bad.
- Dataset: instances of (X, Y) pairs:

编号	色泽	根蒂	敲声	好瓜
1	青绿	蜷缩	浊响	是
2	乌黑	蜷缩	浊响	是
3	青绿	硬挺	清脆	否
4	乌黑	稍蜷	沉闷	否



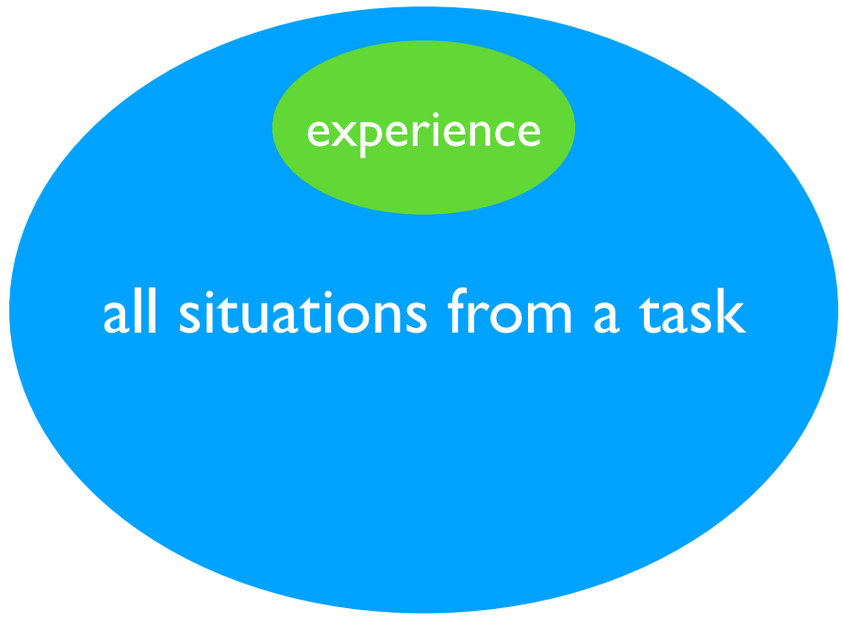
The target is to find the best f^* which has the highest classification accuracy.

Core of Machine Learning: Generalization

- In machine learning, we assume that all the data are drawn from an unknown distribution $\mathcal{D}$. Both training and testing data are drawn from it.

- Training performance:

$$\hat{R}(f) = \frac{1}{N}\sum_{i=1}^{N} Err(f(x_i), y_i)$$



- Testing performance:

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[Err(f(x), y)]$$

In machine learning, we only observe training data, so we can only control training performance.
But actually we want to control the testing performance!
Generalization is to make training and testing performance close.

Bias & Variance Trade-Off

- For linear regression, the expected testing performance is:

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^N}[R(\mathbf{w})] = \mathbb{E}_{(x,y)\sim\mathcal{D},\mathcal{S}\sim\mathcal{D}^N}[(\mathbf{w}^T\mathbf{x} - y)^2]$$

- Bias-Variance decomposition: $\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^N}[R(\mathbf{w})]$ can be decomposed as

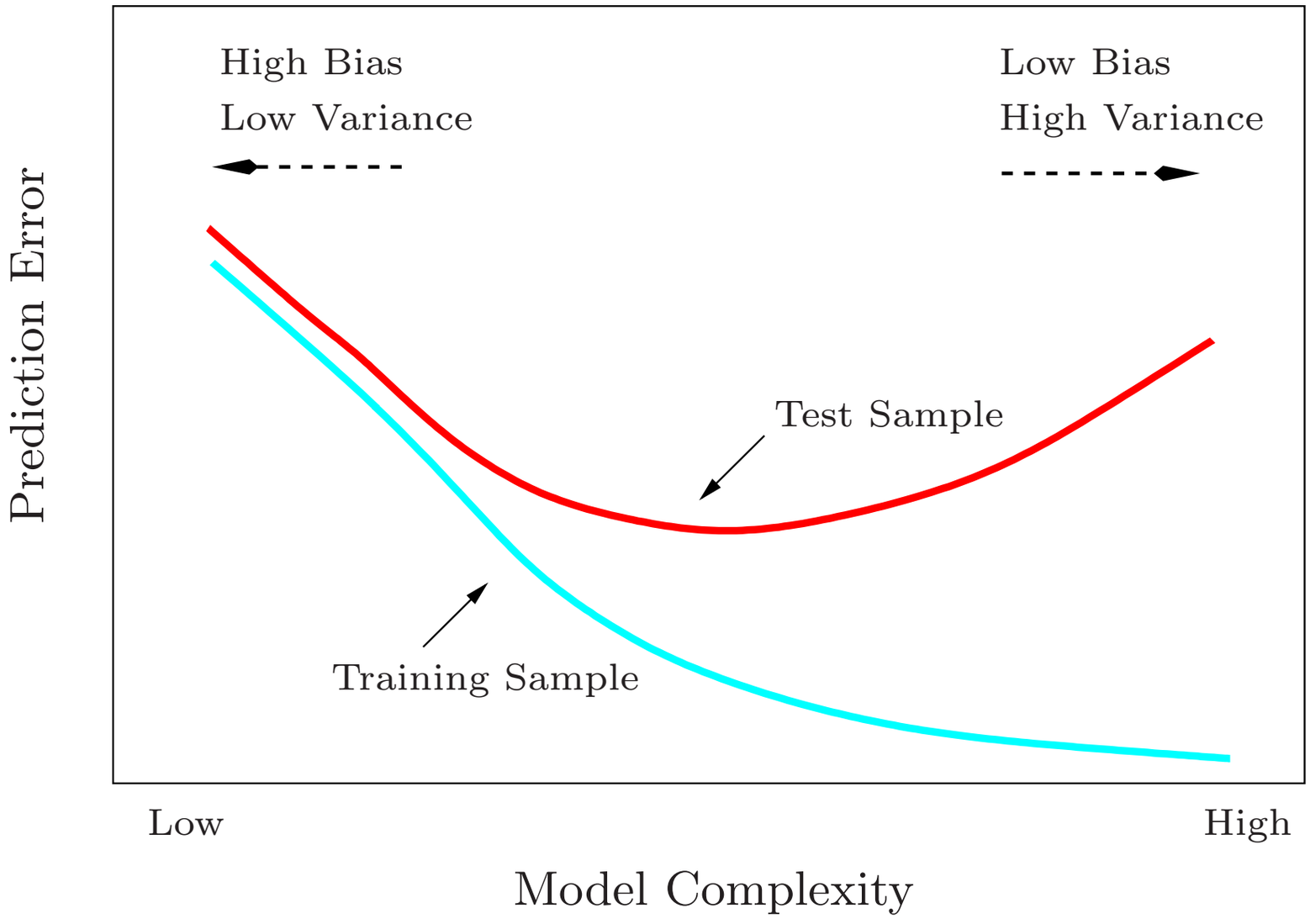
$$\underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^N}[(\mathbf{w}^T\mathbf{x} - \mathbb{E}_{\mathcal{S}\sim\mathcal{D}^N}[\mathbf{w}^T\mathbf{x}])^2]\right]}_{\text{variance: deviation from the mean}} + \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[(\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^N}[\mathbf{w}^T\mathbf{x}] - y)^2\right]}_{\text{bias: error of the mean to the true target}}$$

Model with high variance: The result is not stable, e.g. 1-nearest neighbor.
Model with high bias: The model is not flexible enough, e.g. linear regression.

Bias & Variance Trade-Off

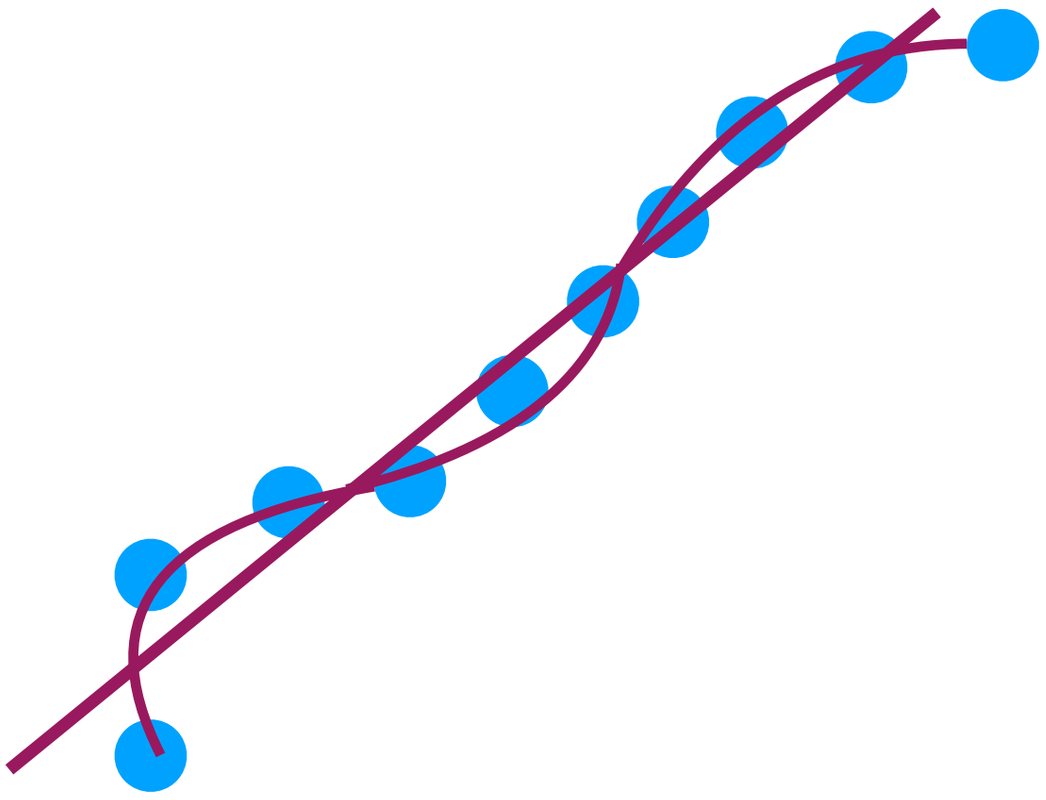
the training & testing performances are close

the training error is low



Need to find the perfect balance!

Occam's Razor



OCCAM'S RAZOR

Anselm BLUMER *
Department of Mathematics and Computer Science, University of Denver, Denver, CO 80208, U.S.A.

Andrzej EHRENFEUCHT **
Department of Computer Science, University of Colorado, Boulder, CO 80302, U.S.A.

David HAUSSLER *
Department of Mathematics and Computer Science, University of Denver, Denver, CO 80208, U.S.A.

Manfred K. WARMUTH ***
Department of Computer and Information Sciences, University of California, Santa Cruz, CA 95064, U.S.A.

Communicated by M.A. Harrison
Received 21 February 1986
Revised 18 July 1986

Occam's Razor

Carl Edward Rasmussen
Department of Mathematical Modelling
Technical University of Denmark
Building 321, DK-2800 Kongens Lyngby, Denmark
carl@imm.dtu.dk http://bayes.imm.dtu.dk

Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square, London WC1N 3AR, England
zoubin@gatsby.ucl.ac.uk http://www.gatsby.ucl.ac.uk

Model complexity should be just enough to perform well on training data, but should not be larger than necessary!

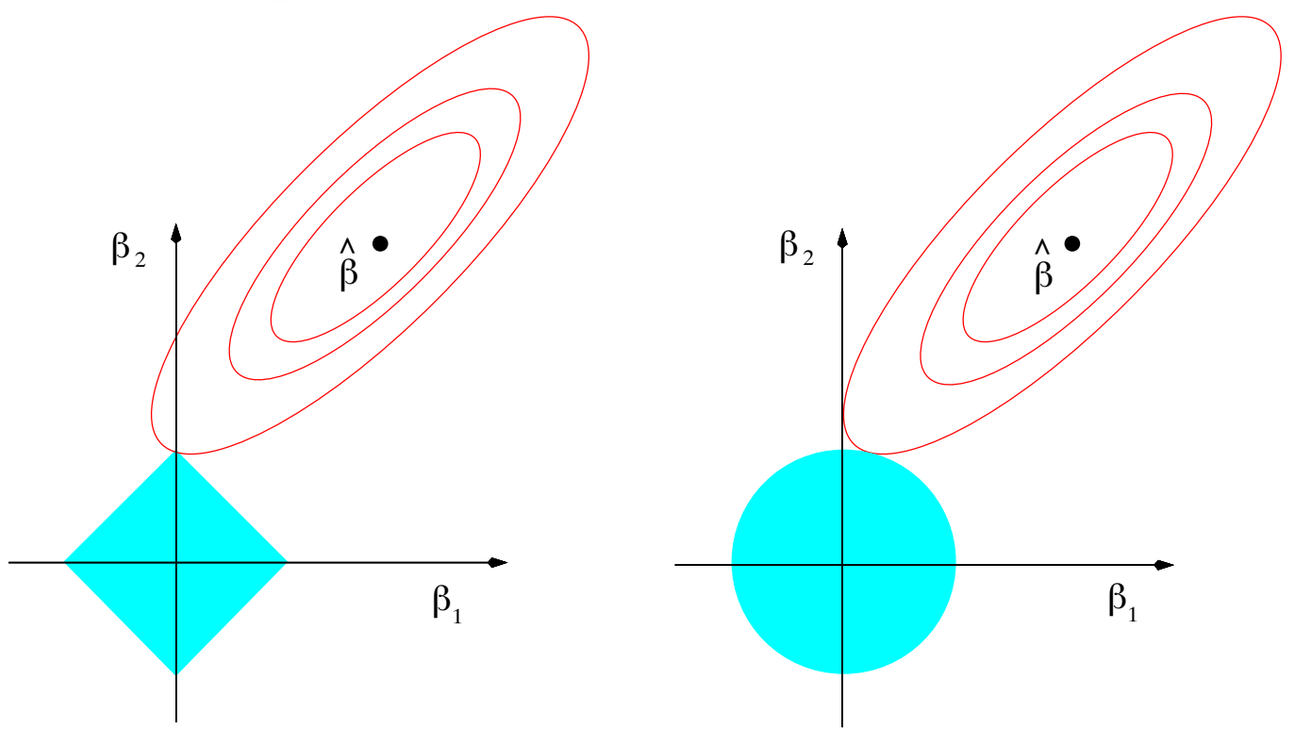
Linear Model & Mean-Squared Error

- Linear model: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$, $\mathbf{w}, \mathbf{x}, \mathbf{b}$ are vectors of dim. d
- Note: equivalent model $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ in dim. $d+1$ since we can then set $\mathbf{x} \to [x_1, x_2, ..., x_d, 1]$ and then hide $\mathbf{b}$.
- Classification rule: $f(x) > 0$ then predict positive, $f(x) < 0$ negative.
- Basic idea: Find the model performs good under the dataset.
- Observation: $\mathbb{I}[f(x) = y]$ is hard to optimize directly.
- Alternative: mean-squared error (MSE): $(f(x) - y)^2$

Linear Regression with Regularization

- Ridge regression: $\hat{R}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$
- LASSO: $\hat{R}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1$

Regularization terms make weights small

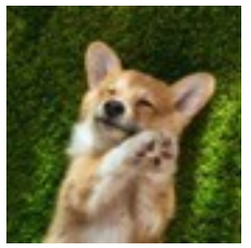


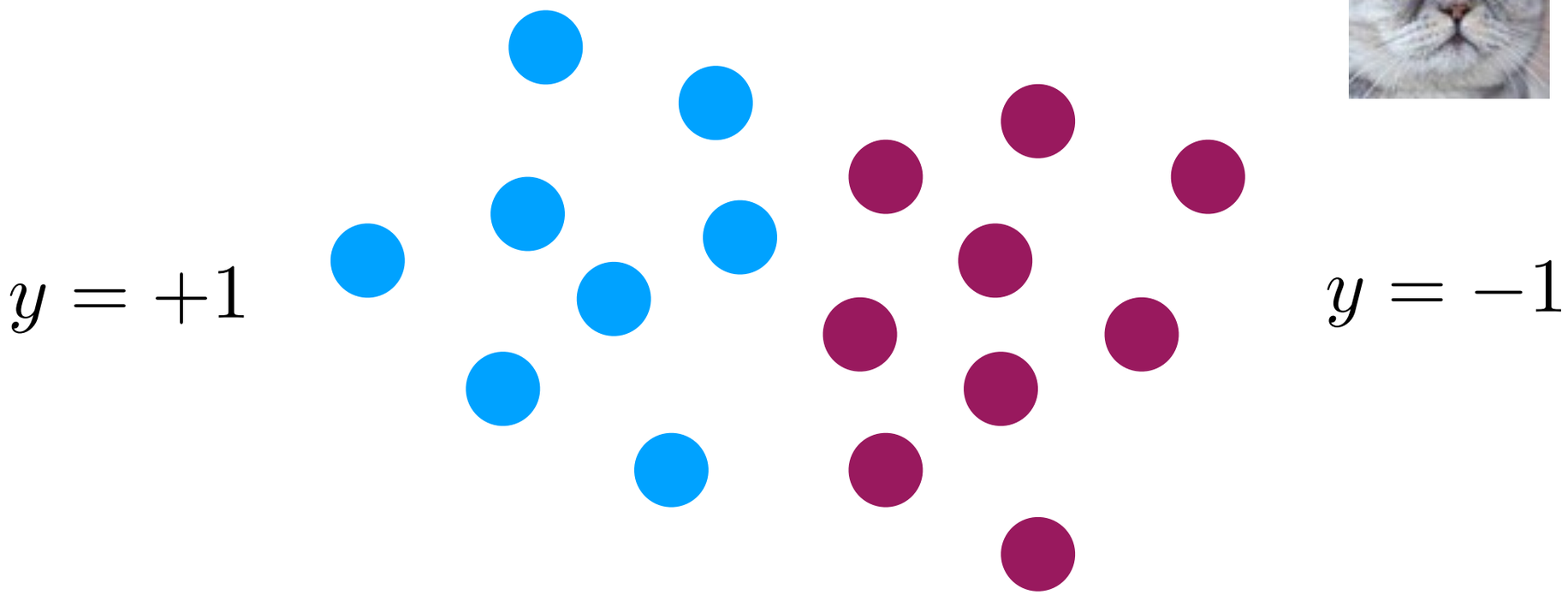
Do linear regression in high dimension and use regularization to control model complexity.
We will learn a strong regularization method in today's lecture!

Binary Classification

- Task: learn $f : X \to Y$ with $Y = \{-1, 1\}$
- Performance measure: correctness of classification $\mathbb{I}[f(x) = y]$
- Dataset: $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$

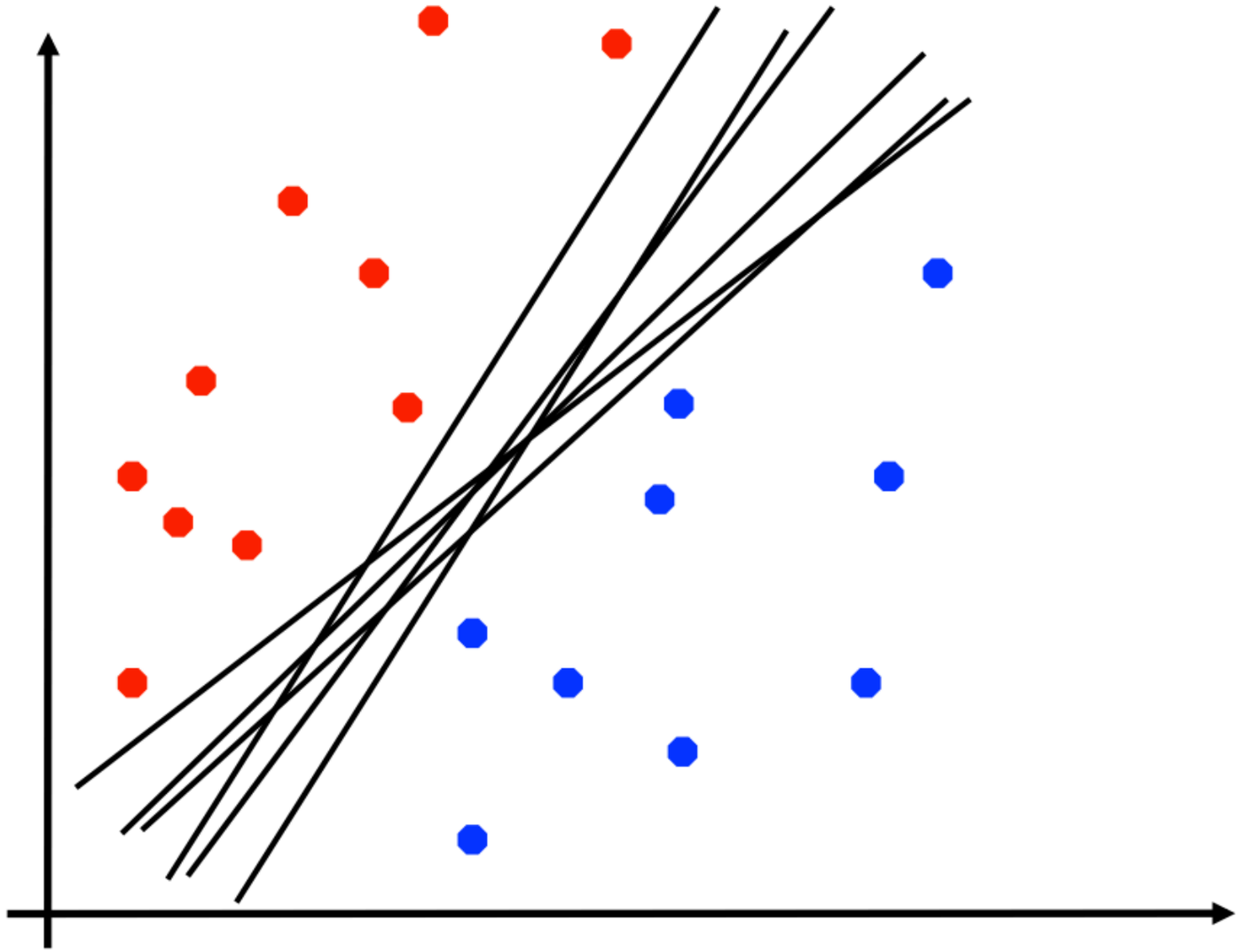
vs.





Linear-separable case:
There exists a linear model that can perfectly separate the classes.

Optimal Linear Separator?

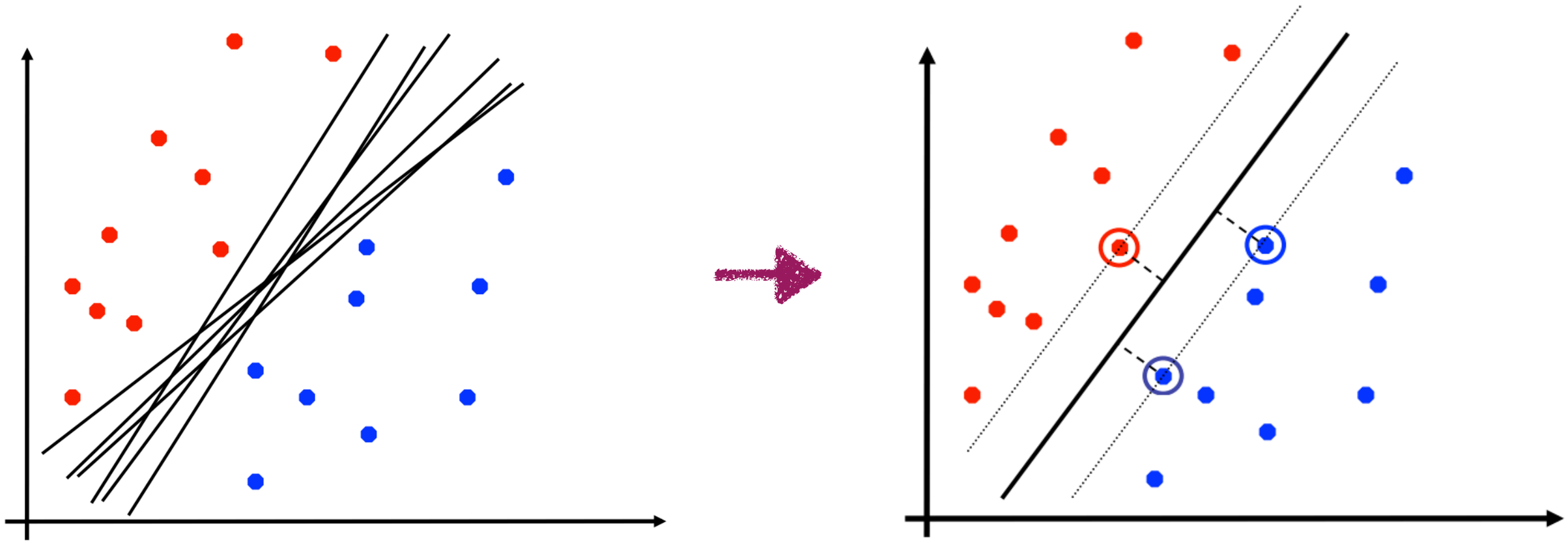


Do all these linear separators have the same testing performance?
If not, how to choose the optimal one?

Machine Learning: II

- Support vector machine (SVM)
  - Linear separable case
  - Linear inseparable case
- AdaBoost
- Take-home messages

Optimal Linear Separator?



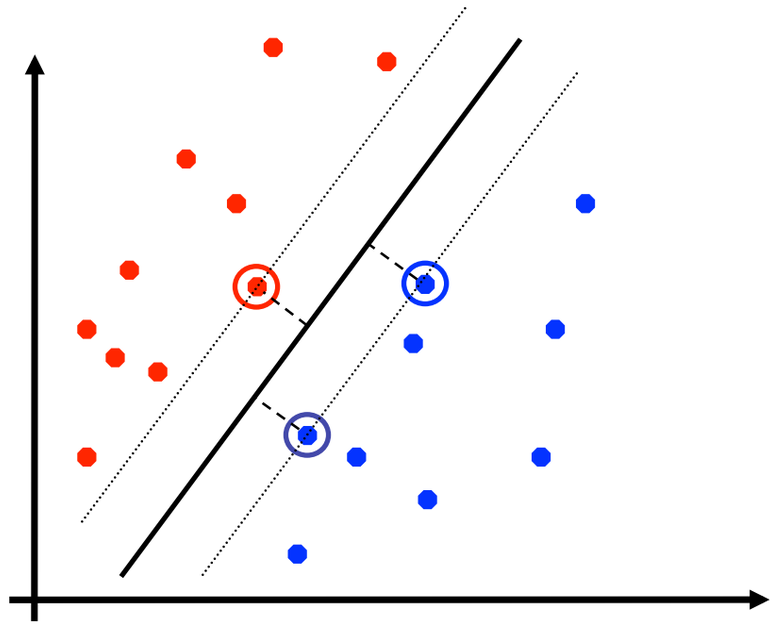
Find the hyperplane exactly on the "midway" in between the two classes.
Why?

If testing data gets closer to the opposite class than any training data,
this hyperplane may still be able to classify it correctly.
Robust to outliers! A form of regularization.

Support Vector Machine (SVM)

- SVMs (Vapnik, 1990's) choose the linear separator with the **largest margin**

Robust to outliers!



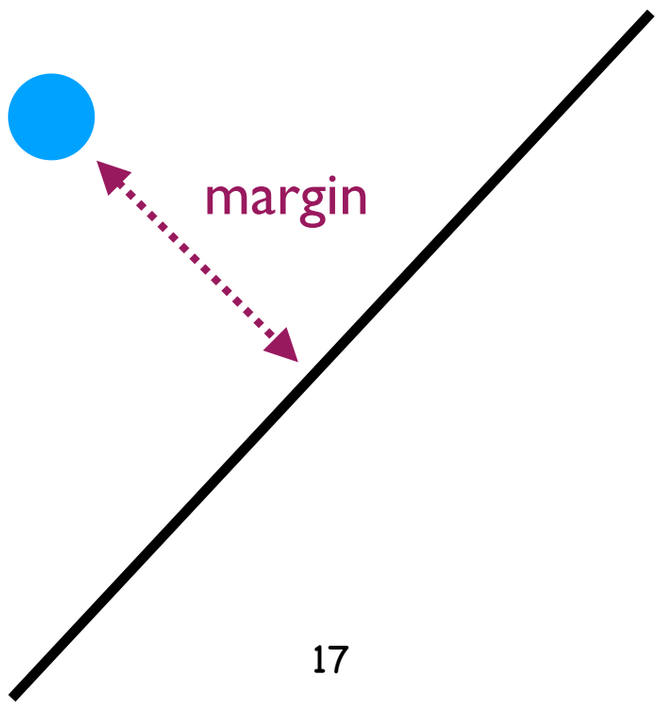
V. Vapnik

- Good according to intuition, theory, practice
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

Margin of Linear Hyperplanes

- The margin of an instance $\mathbf{x}$ to a hyperplane $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$ is the perpendicular distance from the instance to the hyperplane:

$$\gamma(\mathbf{x}) = |\mathbf{w}^T\mathbf{x} + \mathbf{b}|$$

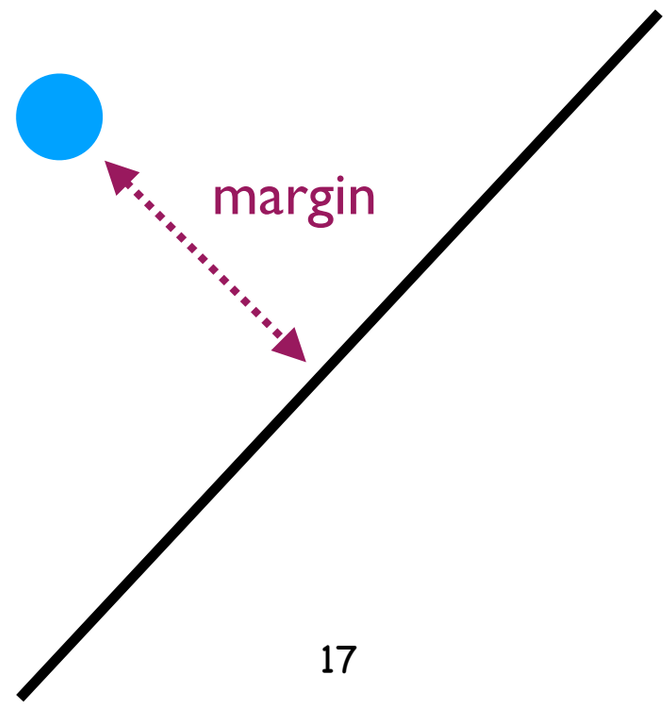


Margin of Linear Hyperplanes

- The margin of an instance $\mathbf{x}$ to a hyperplane $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$ is the perpendicular distance from the instance to the hyperplane:

$$\gamma(\mathbf{x}) = |\mathbf{w}^T\mathbf{x} + \mathbf{b}|$$

- For any $K > 0$, $f(\mathbf{x}) = K(\mathbf{w}^T\mathbf{x} + \mathbf{b})$ has the same classification result to $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$.



Margin of Linear Hyperplanes

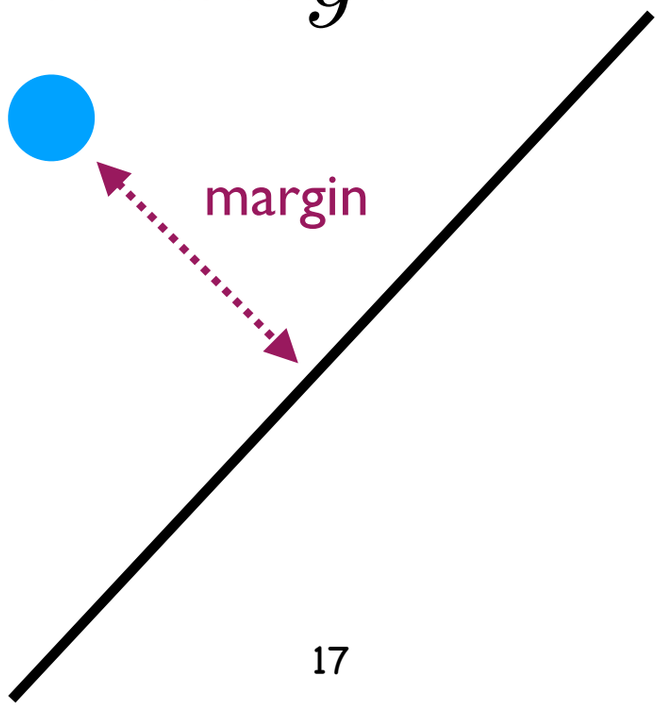
- The margin of an instance $\mathbf{x}$ to a hyperplane $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$ is the perpendicular distance from the instance to the hyperplane:

$$\gamma(\mathbf{x}) = |\mathbf{w}^T\mathbf{x} + \mathbf{b}|$$

- For any $K > 0$, $f(\mathbf{x}) = K(\mathbf{w}^T\mathbf{x} + \mathbf{b})$ has the same classification result to $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$.

- The margin can be normalized and signed w.r.t. the true label y:

$$\gamma(\mathbf{x}) = \frac{y(\mathbf{w}^T\mathbf{x} + \mathbf{b})}{\|\mathbf{w}\|}$$



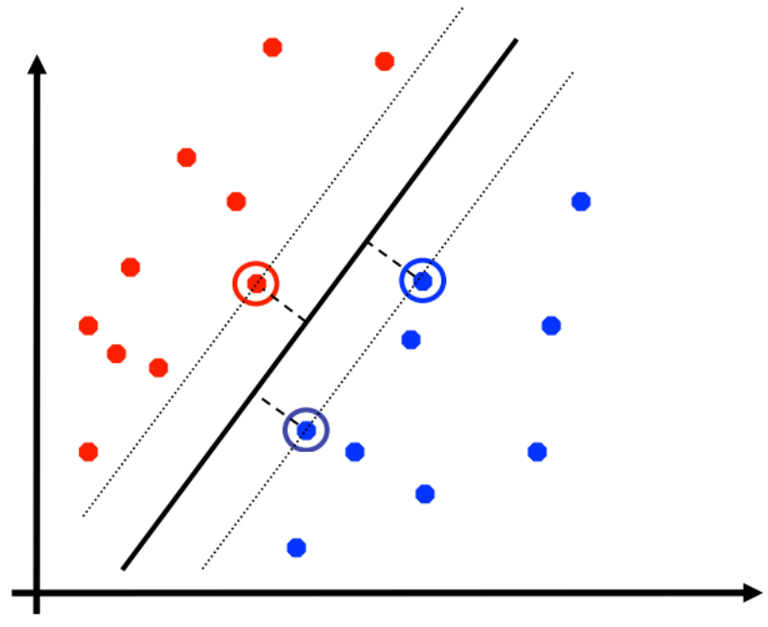
The larger the margin,
the more correct and confident the prediction.

Margin Maximization

- The (signed) margin of an instance: $\gamma(\mathbf{x}) = \dfrac{y(\mathbf{w}^T\mathbf{x} + \mathbf{b})}{\|\mathbf{w}\|}$
- Want to maximize the margins for all training data:

$$\max_{\mathbf{w},\mathbf{b}} \gamma$$

$$s.t. \quad \frac{y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})}{\|\mathbf{w}\|} \geq \gamma, i \in 1, ..., N$$



Margin Maximization

- The (signed) margin of an instance: $\gamma(\mathbf{x}) = \dfrac{y(\mathbf{w}^T\mathbf{x} + \mathbf{b})}{\|\mathbf{w}\|}$
- Want to maximize the margins for all training data:

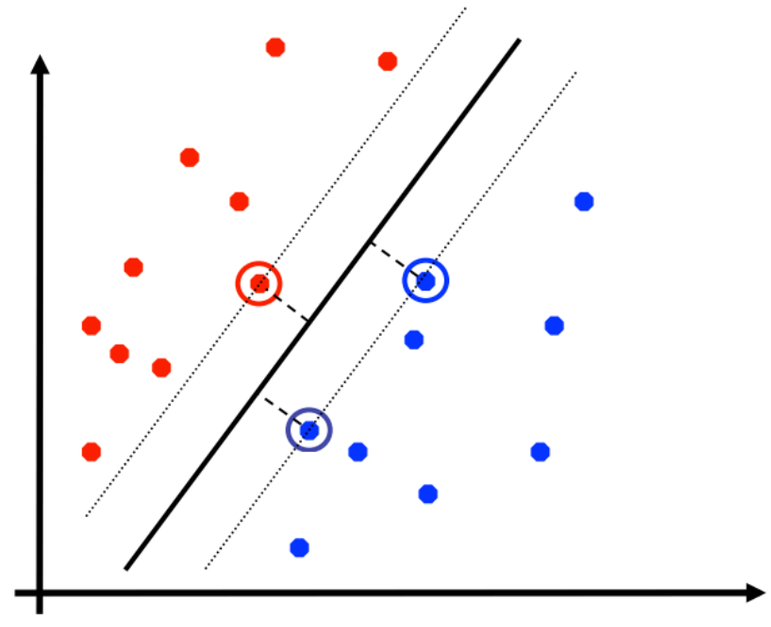
$$\max_{\mathbf{w},\mathbf{b}} \gamma$$

$$s.t. \quad \frac{y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})}{\|\mathbf{w}\|} \geq \gamma, i \in 1, ..., N$$

- For easiness of optimization, turn into:

$$\max_{\mathbf{w},\mathbf{b}} \frac{1}{\|\mathbf{w}\|}$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$



Margin Maximization

- The (signed) margin of an instance: $\gamma(\mathbf{x}) = \dfrac{y(\mathbf{w}^T\mathbf{x} + \mathbf{b})}{\|\mathbf{w}\|}$
- Want to maximize the margins for all training data:

$$\max_{\mathbf{w},\mathbf{b}} \gamma$$

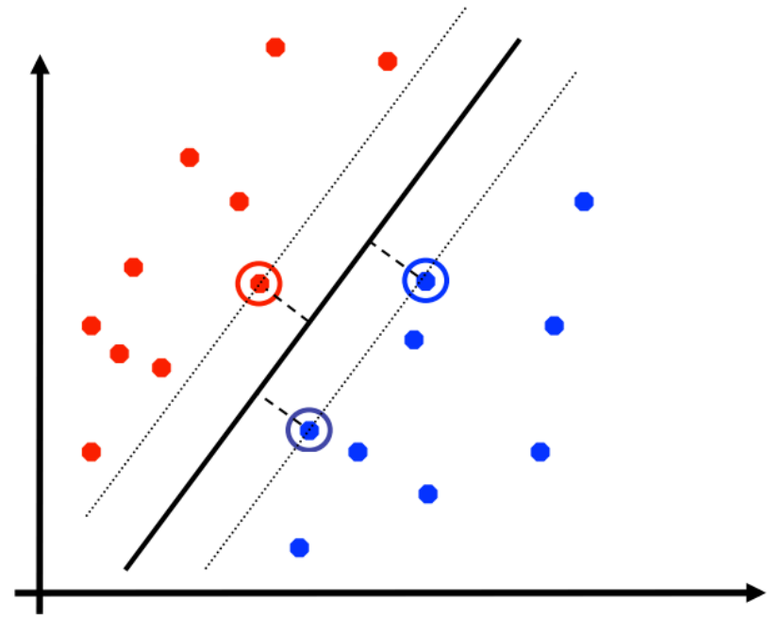
$$s.t. \quad \frac{y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})}{\|\mathbf{w}\|} \geq \gamma, i \in 1, ..., N$$

- For easiness of optimization, turn into:

$$\max_{\mathbf{w},\mathbf{b}} \frac{1}{\|\mathbf{w}\|}$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

Why these two optimization problems are equivalent?



Hard-Margin SVM

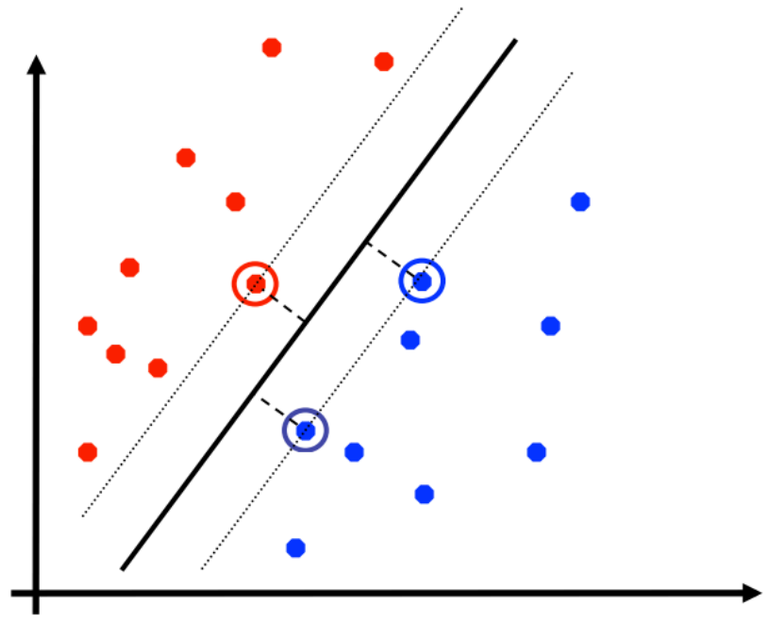
- For easiness of optimization, turn right into below:

$$\max_{\mathbf{w},\mathbf{b}} \frac{1}{\|\mathbf{w}\|}$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$



Hard-Margin SVM

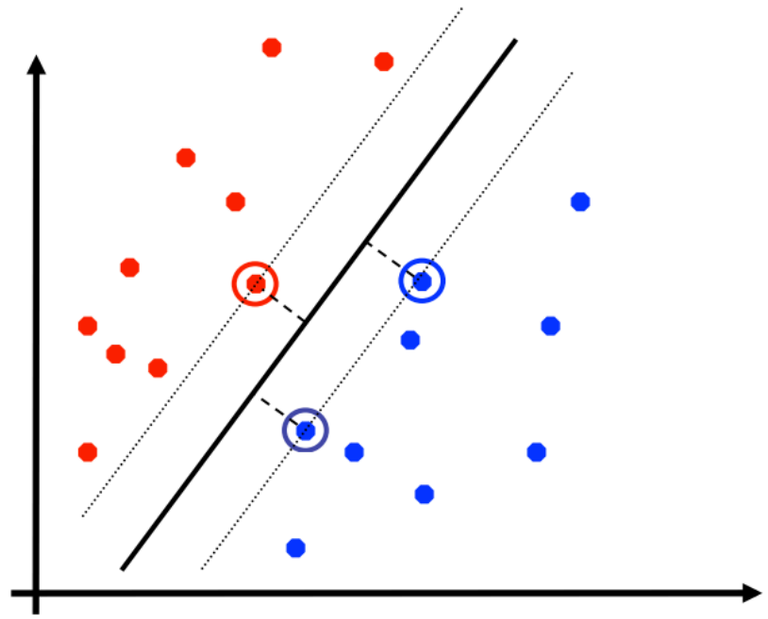
- For easiness of optimization, turn right into below:

$$\max_{\mathbf{w},\mathbf{b}} \frac{1}{\|\mathbf{w}\|}$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

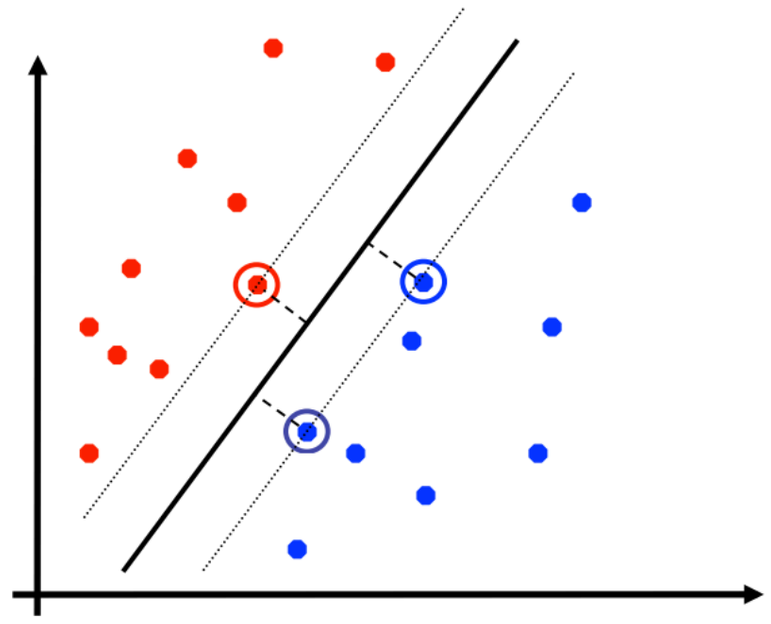


SVM optimization objective for linear-separable problems.
How to solve this constrained optimization problem?

Solve for Hard-Margin SVM

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$



- Basic idea: eliminate the constraints.

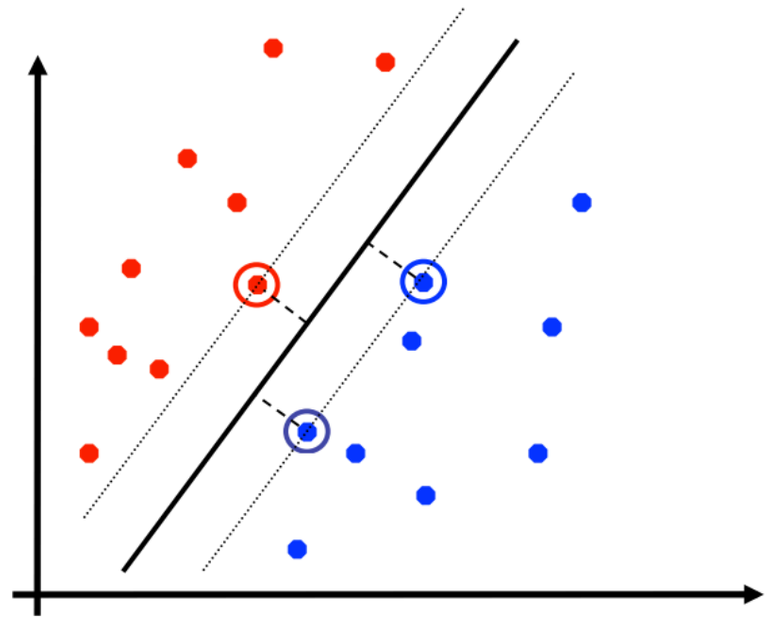
- Lagrange multiplier method: introduce the adversary parameters λ_i

$$\min_{\mathbf{w},\mathbf{b}} \max_{\lambda_i \geq 0} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})]\right]$$

Solve for Hard-Margin SVM

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$



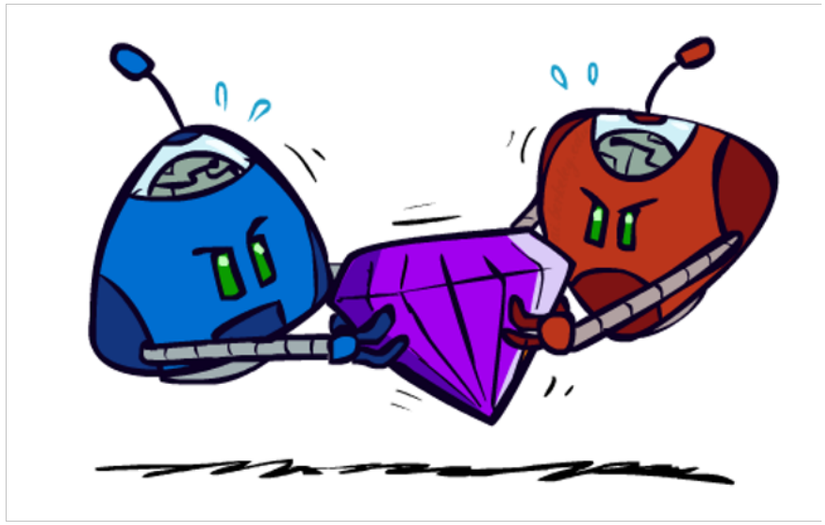
- Basic idea: eliminate the constraints.

- Lagrange multiplier method: introduce the adversary parameters λ_i

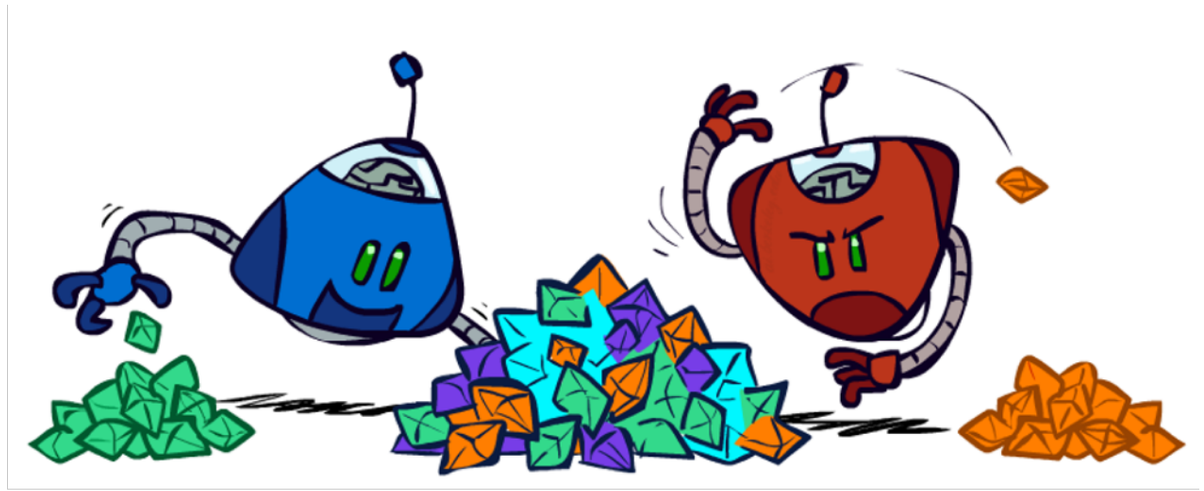
$$\min_{\mathbf{w},\mathbf{b}} \max_{\lambda_i \geq 0} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})]\right]$$

Are you familiar with this problem?

Two-Player Zero-Sum Game



vs.



In zero-sum games, the utility functions of the two players are coupled:
How much one wins equals to how much the other loses: competitive.
Can the players be cooperative?

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j

zero-sum

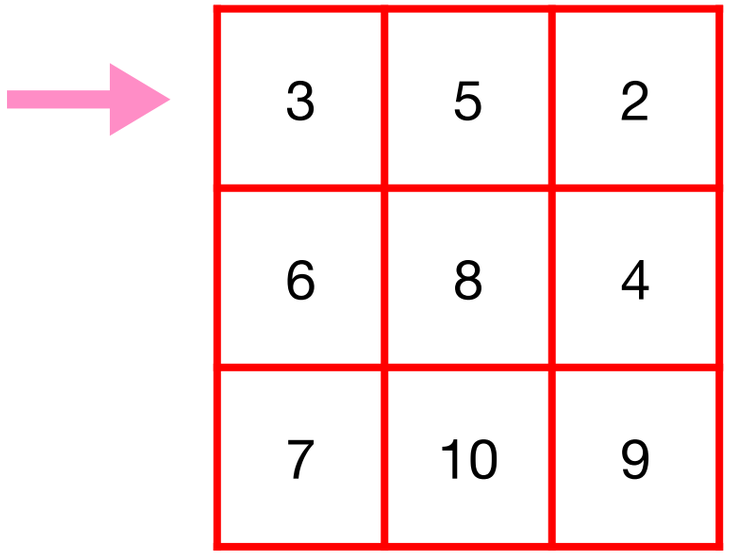
- A MinMax Game.

3	5	2
6	8	4
7	10	9

$$U(i^*, j^*) = \min_i \max_j U(i, j)$$

The Simplest Formulation: Two-Step Game

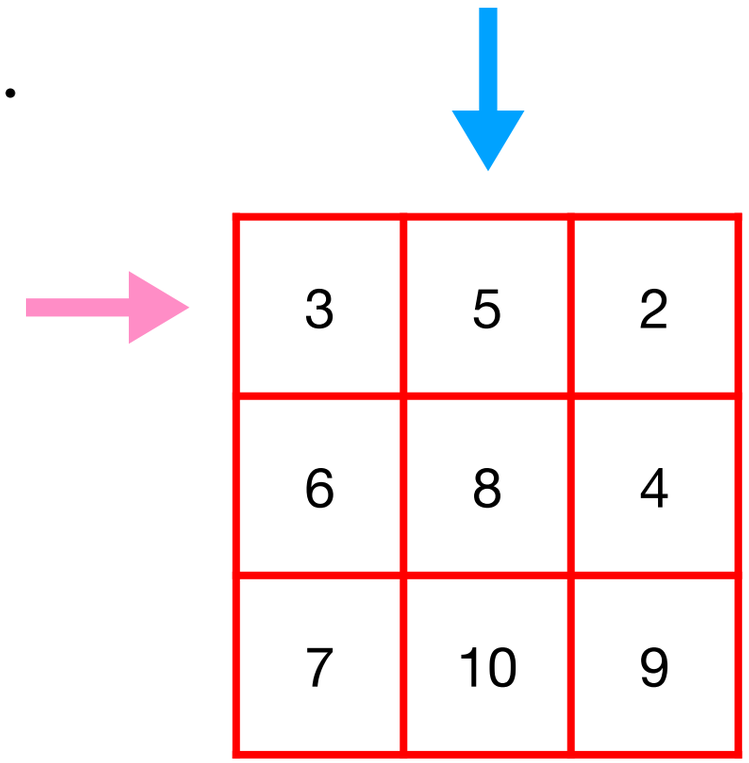
- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j zero-sum
- A MinMax Game.



$$U(i^*, j^*) = \min_i \max_j U(i, j)$$

The Simplest Formulation: Two-Step Game

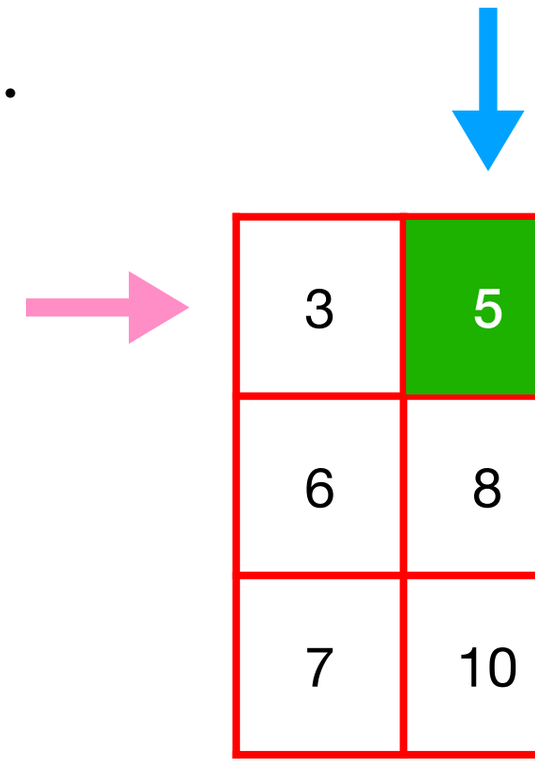
- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j zero-sum
- A MinMax Game.



$$U(i^*, j^*) = \min_i \max_j U(i, j)$$

The Simplest Formulation: Two-Step Game

- Two step game: Alice chooses row i , then Bob chooses column j
- Outcome: Alice loses (Bob wins) the utility in entry i, j zero-sum
- A MinMax Game.



$$U(i^*, j^*) = \min_i \max_j U(i, j)$$

The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

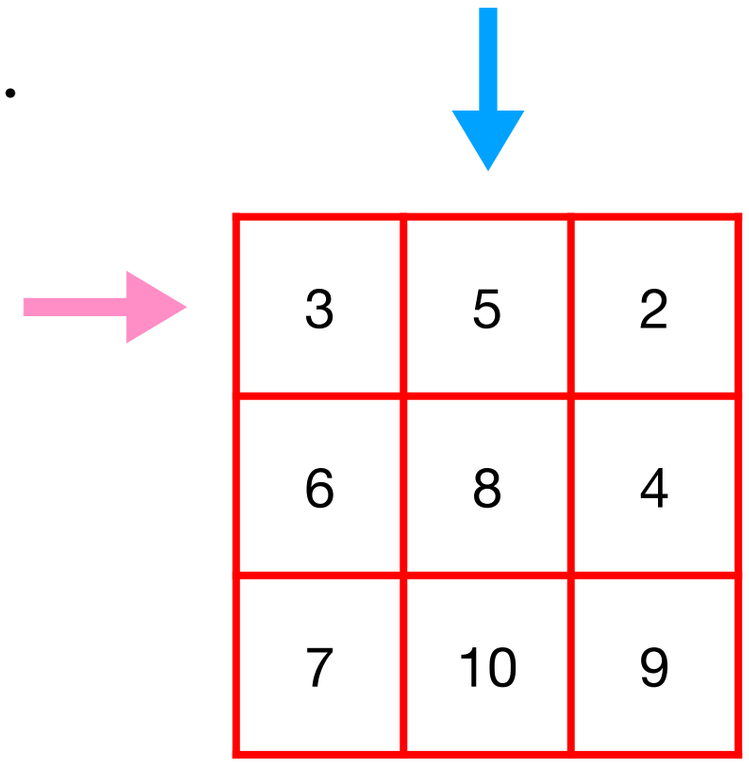
The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.

3	5	2
6	8	4
7	10	9

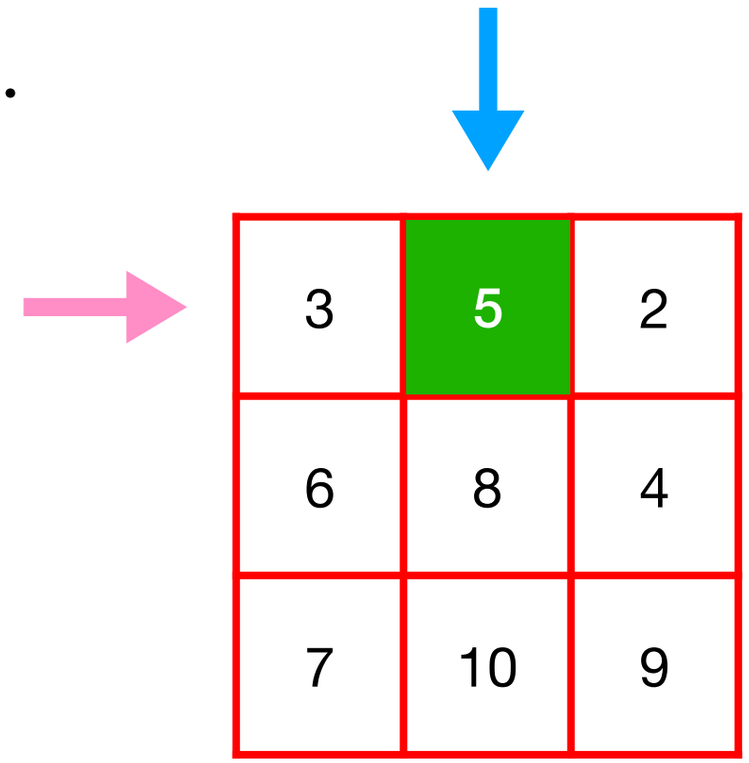
The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.



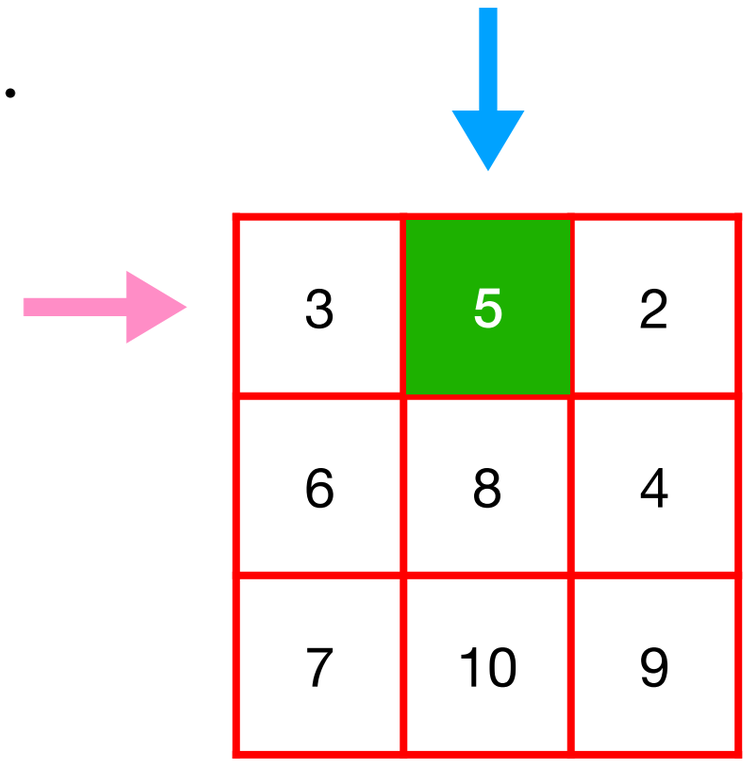
The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.



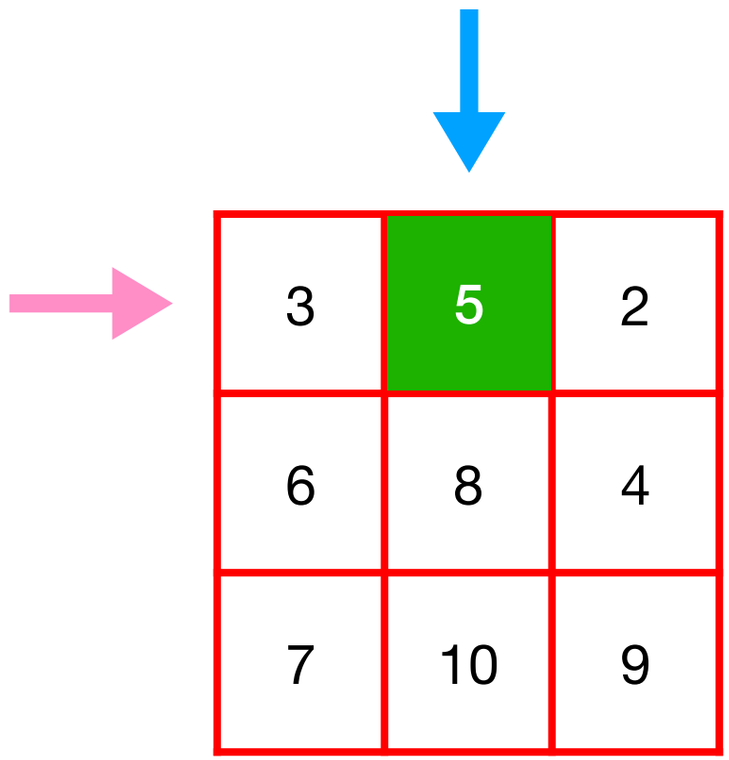
The Simplest Formulation: Two-Step Game

- Two step game: Bob chooses row j, then Alice chooses column i
- Outcome: Alice loses (Bob wins) the utility in entry i, j
- A MaxMin Game.



$$U(i^*, j^*) = \max_j \min_i U(i, j)$$

Is MinMax equivalent to MaxMin?



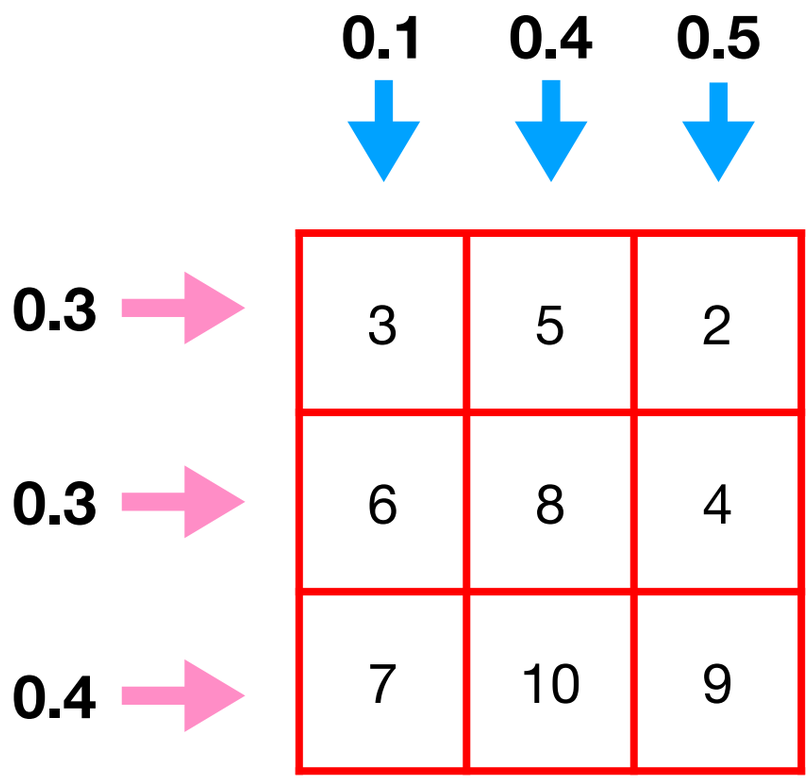
Does Alice (Bob) lose (win) the same utility in MinMax and MaxMin games?

Theorem:

$$\max_j \min_i U(i,j) \leq \min_i \max_j U(i,j)$$

Moving first is always worse!

Deterministic vs. Stochastic Strategy



Mixed (Stochastic) strategy: choosing a distribution over actions instead of a single action (pure or deterministic strategy)

Von Neumann's Minimax Theorem:

$$\min_{P_i} \max_{P_j} \mathbb{E}_{P_i,P_j}\left[U(i,j)\right] = \max_{P_j} \min_{P_i} \mathbb{E}_{P_i,P_j}\left[U(i,j)\right]$$

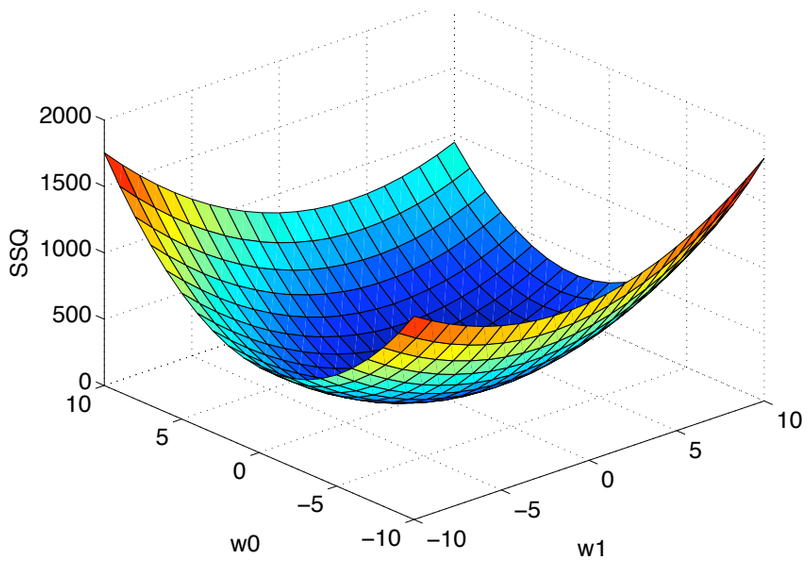
MinMax and MaxMin Games are equivalent for mixed strategy!

Nash Equilibrium

$$\min_{P_i} \max_{P_j} \mathbb{E}_{P_i,P_j}\left[U(i,j)\right] = \max_{P_j} \min_{P_i} \mathbb{E}_{P_i,P_j}\left[U(i,j)\right]$$

- Strong duality: the solution pair P_i^*, P_j^* achieving this equation is called the saddle point of the two-step zero-sum game.
- This is also the Nash equilibrium of the game.

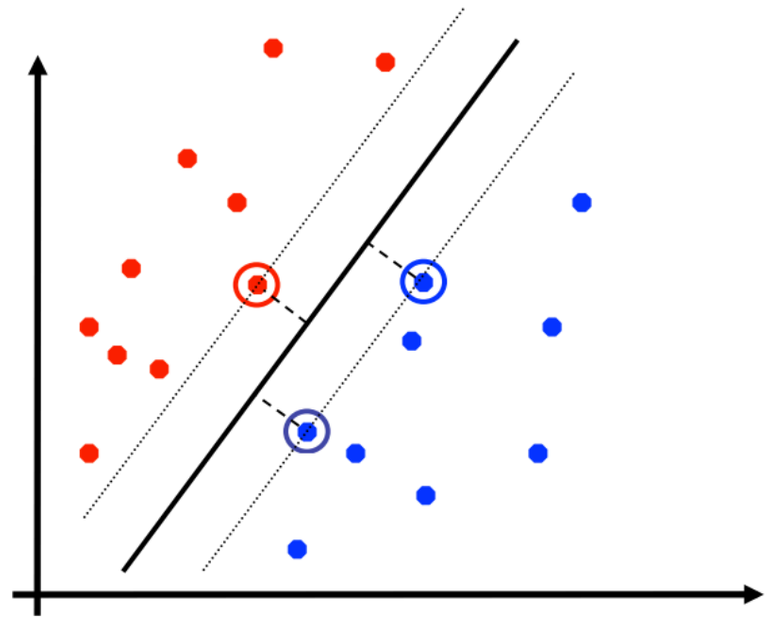
The strong duality holds even for deterministic functions when the utility function is convex for minimizer and concave for maximizer!



Strong Duality in SVM

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$



- Convex problem. Strong duality for Lagrange multiplier solution holds!

- We can interchange the Min Max operators.

$$\min_{\mathbf{w},\mathbf{b}} \max_{\lambda_i \geq 0} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})]\right]$$

$$\Downarrow$$

$$\max_{\lambda_i \geq 0} \min_{\mathbf{w},\mathbf{b}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})]\right]$$

The Dual Problem

$$\max_{\lambda_i \geq 0} \min_{\mathbf{w},\mathbf{b}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + \mathbf{b})]\right]$$

- Suppose that λ_i are fixed, then we can directly solve for $\mathbf{w}, \mathbf{b}$ by taking derivatives:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \lambda_i y_i = 0$$

We can do this because Max is done first, then the value of λ_i are fixed.

The Dual Problem

$$\max_{\lambda_i \geq 0} \min_{\mathbf{w},\mathbf{b}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + \mathbf{b})]\right]$$

- Suppose that λ_i are fixed, then we can directly solve for $\mathbf{w}, \mathbf{b}$ by taking derivatives:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \lambda_i y_i = 0$$

We can do this because Max is done first, then the value of λ_i are fixed.

- Put the above results back, we transform the problem into

$$\max_{\lambda_i} \left[\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T\right]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

Complementary Slackness

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

- When λ_i are fixed, then we can obtain $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$

How to obtain $\mathbf{b}$?

- From the dual problem: $\max_{\lambda_i \geq 0} \min_{\mathbf{w},\mathbf{b}} [\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + \mathbf{b})]]$
- We observe that $\forall i, \quad \lambda_i [1 - y_i(\mathbf{w}^T \mathbf{x}_i + \mathbf{b})] = 0$ Why?
- Furthermore, there exists non-zero λ_i , we can obtain $\mathbf{b}$ from it.

The KKT Condition

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

primal problem

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

dual problem

- The solution of the dual problem: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

The KKT Condition

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

primal problem

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

dual problem

- The solution of the dual problem: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

- From the constraints of the primal & dual problems, we have three observations:
  - Primal feasibility: $y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1$
  - Dual feasibility: $\lambda_i \geq 0$
  - Complementary slackness: $\forall i, \quad \lambda_i[1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})] = 0$

The KKT Condition

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

primal problem

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

dual problem

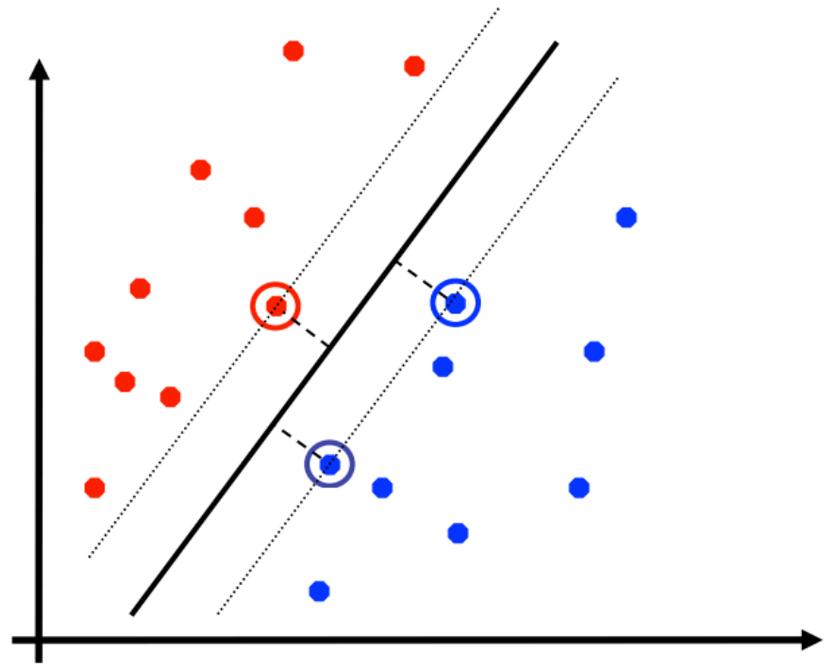
- The solution of the dual problem: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

- From the constraints of the primal & dual problems, we have three observations:
  - Primal feasibility: $y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1$
  - Dual feasibility: $\lambda_i \geq 0$
  - Complementary slackness: $\forall i, \quad \lambda_i[1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})] = 0$

These three conditions are called the Karush–Kuhn–Tucker conditions.

Support Vectors

- The complementary slackness condition: $\forall i, \quad \lambda_i[1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})] = 0$
- When $\lambda_i \neq 0$, we have $1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) = 0$
- $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$

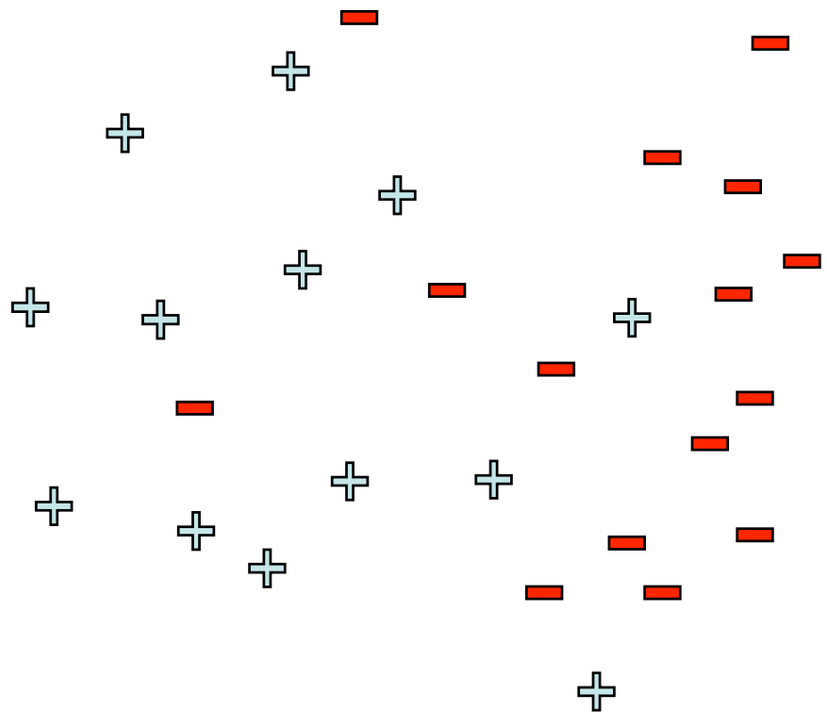


The model weights are formulated by instances exactly on the margins, which are called support vectors.
This is where the name SVM comes from!

Machine Learning: II

- Support vector machine (SVM)
  - Linear separable case
  - Linear inseparable case
- AdaBoost
- Take-home messages

Linear Inseparable Case?



vs.

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

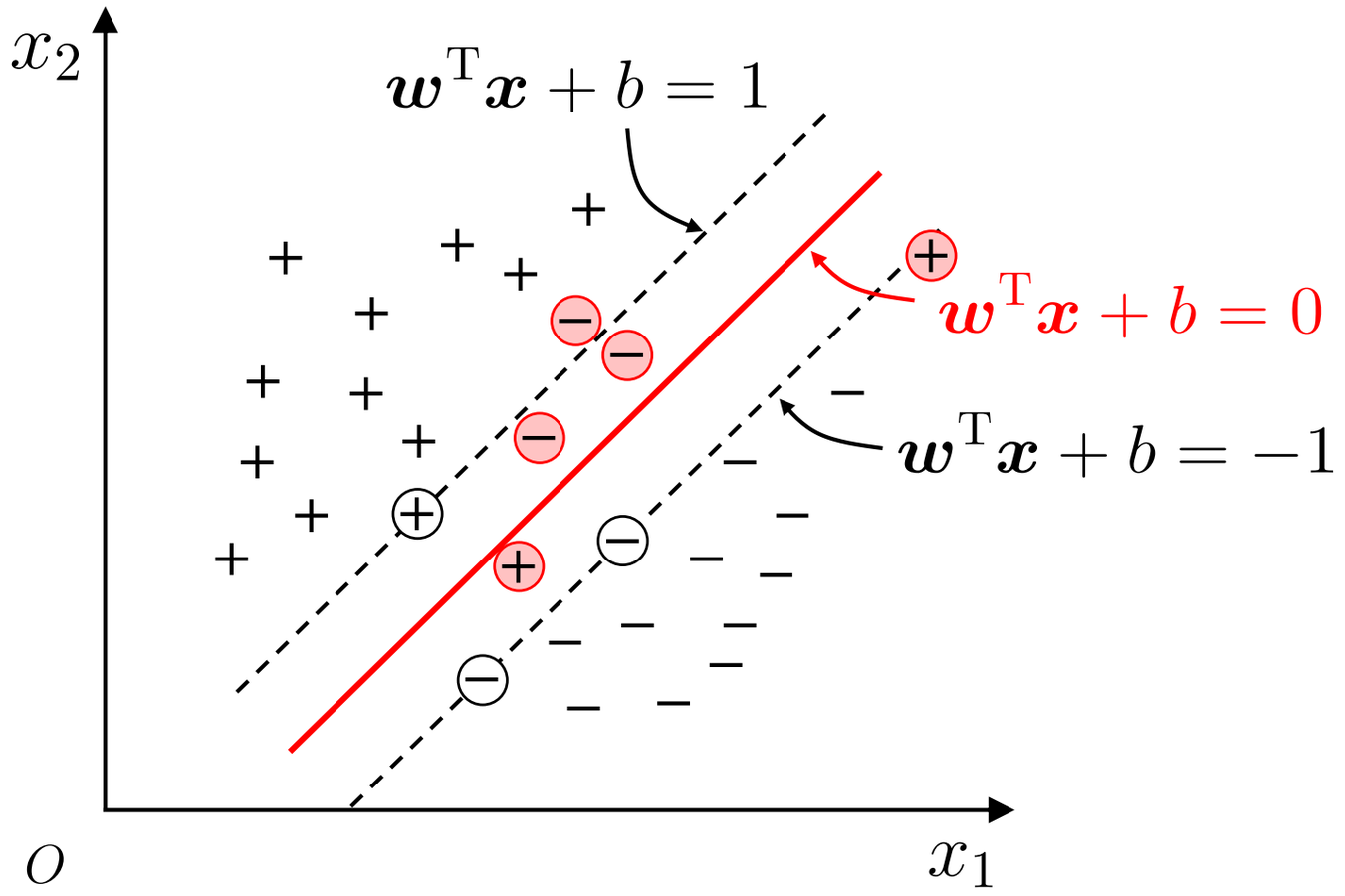
Most real-world data are not linear separable.
No solution exists when the margin constraints are all satisfied.

Soft-Margin SVM

- Idea one: relax the margin constraints by allowing small violations:

$$\min_{\mathbf{w},\mathbf{b},\xi_i} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1 - \xi_i, \xi_i \geq 0, i \in 1, ..., N$$



Hinge Loss

$$\min_{\mathbf{w},\mathbf{b},\xi_i} \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1 - \xi_i, \xi_i \geq 0, i \in 1, ..., N$$

- Since we allow small errors ξ_i , we only want $1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})$ to approach zero, so we can use the following objective instead, in which an constant $C > 0$ is introduced.

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})\right)$$

Hinge Loss

$$\min_{\mathbf{w},\mathbf{b},\xi_i} \frac{1}{2}\|\mathbf{w}\|^2$$
$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1 - \xi_i, \xi_i \geq 0, i \in 1, ..., N$$

- Since we allow small errors ξ_i, we only want $1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})$ to approach zero, so we can use the following objective instead, in which an constant $C > 0$ is introduced.

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})\right)$$

The first term is the regularization term similar to ridge regression.
The second term is called the hinge loss.
This loss + regularization formulation is the foundation of statistical learning.

Kernel Method

- Idea two: do non-linear transformation to the inputs:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + \mathbf{b}$$

- Example: polynomial transformation:

$$\phi(\mathbf{x}) = [x_1^2\ x_2^2\ \ldots\ x_d^2\ x_1\ x_2\ \ldots\ x_d]$$

General idea: map the input into higher-dimensional linear spaces.
How high can we go? Can we go to infinite-dimensional space?

Kernel Method

- Let's turn to the dual problem:

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

- The dual solution: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

The inner product of inputs appears in both optimization and solution.

Kernel Method

- Let's turn to the dual problem:

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

- The dual solution: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

The inner product of inputs appears in both optimization and solution.

Kernel Method

- Let's turn to the dual problem:

$$\max_{\lambda_i} [\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j^T]$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0, \quad \lambda_i \geq 0, i = 1, ..., N.$$

- The dual solution: $f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + \mathbf{b}$

The inner product of inputs appears in both optimization and solution.

- Kernel functions: $\kappa(\mathbf{x}, \mathbf{x}') = \phi^T(\mathbf{x})\phi(\mathbf{x}')$

There are many kernel functions equivalent to inner product calculation, even for infinite dimensional transformations.
Very powerful tool for transforming linear models to non-linear problems.

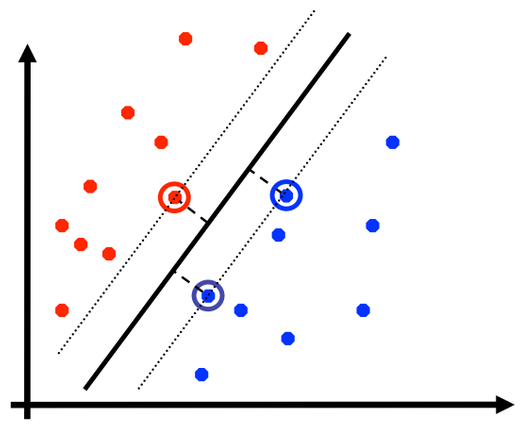
Recap

Support vector machines: 3 key ideas

1. Use **optimization** to find solution (i.e. a hyperplane) with few errors

2. Seek **large margin** separator to improve generalization

3. Use **kernel trick** to make large feature spaces computationally efficient



Machine Learning: II

- Support vector machine (SVM)
  - Linear separable case
  - Linear inseparable case
- AdaBoost
- Take-home messages

Generalized Notion of Margin

- Consider the primal problem of SVM:

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1, i \in 1, ..., N$$

- The margin is $y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b})$

- The generalized notion of margin: if we have a binary classifier $f(\mathbf{x})$ whose output is within $[-1, 1]$, we can define the margin as $yf(\mathbf{x})$

The larger the margin, the more confident the classifier to output the correct label.
Can we borrow the idea of SVM to obtain more powerful classifiers?

Large-Margin Voting Classifiers

- SVM learns the max-margin linear model: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b}$
- We can relax the formulation into $H(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x})$, in which $h_t(\mathbf{x})$ can be any transformation of $\mathbf{x}$.
- In special, we assume $h_t(\mathbf{x})$ to be some base classifiers. Then $H(\mathbf{x})$ is the voting result of the base classifiers.

We want to combine the power of these base classifiers to form a stronger voting classifier.
The key idea lies also on margin maximization.

Boosting

- boosting = general method of converting rough rules of thumb into highly accurate prediction rule
- technically:
  - assume given "weak" learning algorithm that can consistently find classifiers ("rules of thumb") at least slightly better than random, say, accuracy $\geq 55\%$ (in two-class setting) ["weak learning assumption"]
  - given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99%

The History of Boosting

- [Valiant '84]:
  - introduced theoretical ("PAC") model for studying machine learning
- [Kearns & Valiant '88]:
  - open problem of finding a boosting algorithm
  - if boosting possible, then...
    - can use (fairly) wild guesses to produce highly accurate predictions
    - if can learn "part way" then can learn "all the way"
    - should be able to improve any learning algorithm
    - for any learning problem:
      - either can always learn with nearly perfect accuracy
      - or there exist cases where cannot learn even slightly better than random guessing

The History of Boosting

- [Schapire '89]:
  - first provable boosting algorithm
- [Freund '90]:
  - "optimal" algorithm that "boosts by majority"
- [Drucker, Schapire & Simard '92]:
  - first experiments using boosting
  - limited by practical drawbacks
- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms

The Basic Mechanism of Boosting

- Boosting learns voting classifier $H(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x})$ by gradually adding base classifiers $h_t(\mathbf{x})$ into votes.

  - given training set $(x_1, y_1), \ldots, (x_m, y_m)$
  - $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
  - for $t = 1, \ldots, T$:
    - construct distribution D_t on $\{1, \ldots, m\}$
    - find weak classifier ("rule of thumb")

      $$h_t : X \to \{-1, +1\}$$

      with small error ϵ_t on D_t:

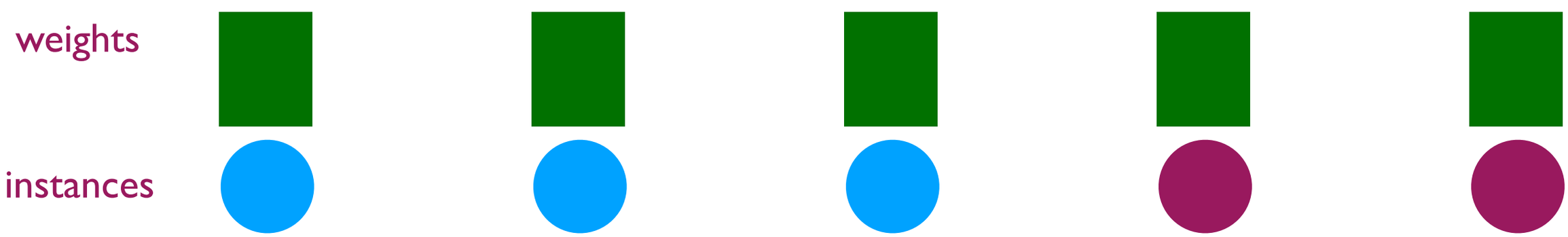
      $$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

  - output final classifier H_{final}

central step of boosting: constructing a distribution on data in each round

The Basic Mechanism of AdaBoost

- In each boosting iteration, learn a new base classifier based on instance distribution weights.

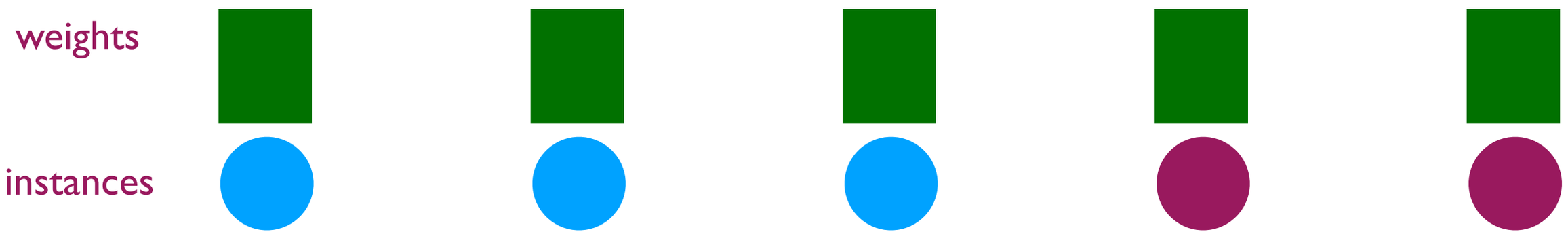


The Basic Mechanism of AdaBoost

- In each boosting iteration, learn a new base classifier based on instance distribution weights.

- Margin calculation:

$$h_1 : y_1 h_1(\mathbf{x}_1) = 1, y_2 h_1(\mathbf{x}_2) = -1, y_3 h_1(\mathbf{x}_3) = 1, y_4 h_1(\mathbf{x}_4) = 1, y_5 h_1(\mathbf{x}_5) = -1$$



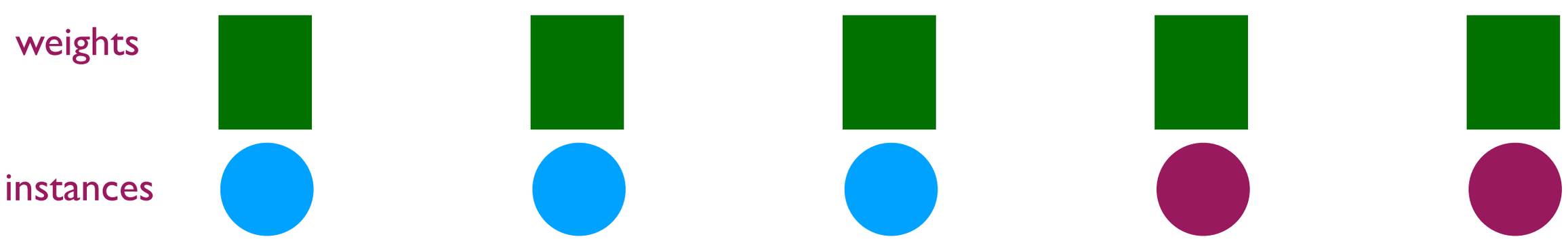
The Basic Mechanism of AdaBoost

- In each boosting iteration, learn a new base classifier based on instance distribution weights.

- Margin calculation:

$$h_1 : y_1 h_1(\mathbf{x}_1) = 1, y_2 h_1(\mathbf{x}_2) = -1, y_3 h_1(\mathbf{x}_3) = 1, y_4 h_1(\mathbf{x}_4) = 1, y_5 h_1(\mathbf{x}_5) = -1$$

- Set weight: margin large then model weight large, otherwise model weight small: want to make more instances to have larger margins.



The Basic Mechanism of AdaBoost

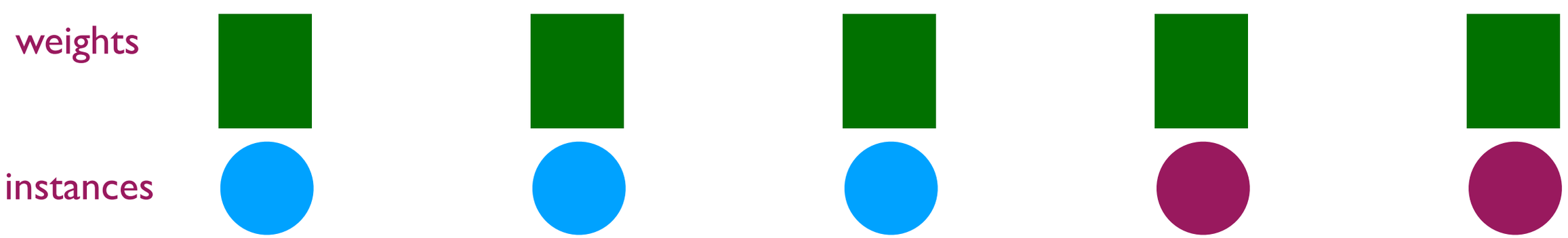
- In each boosting iteration, learn a new base classifier based on instance distribution weights.

- Margin calculation:

$$h_1 : y_1 h_1(\mathbf{x}_1) = 1, y_2 h_1(\mathbf{x}_2) = -1, y_3 h_1(\mathbf{x}_3) = 1, y_4 h_1(\mathbf{x}_4) = 1, y_5 h_1(\mathbf{x}_5) = -1$$

- Set weight: margin large then model weight large, otherwise model weight small: want to make more instances to have larger margins.

- Change data distribution: increase instance weight for small-margin instances.



The Basic Mechanism of AdaBoost

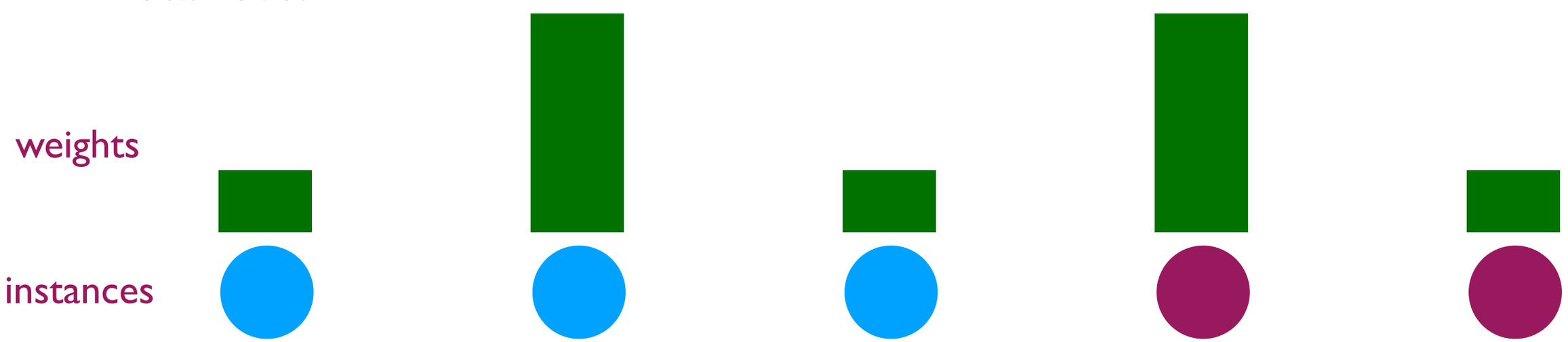
- In each boosting iteration, learn a new base classifier based on instance distribution weights.

- Margin calculation:

$$h_1 : y_1 h_1(\mathbf{x}_1) = 1, y_2 h_1(\mathbf{x}_2) = -1, y_3 h_1(\mathbf{x}_3) = 1, y_4 h_1(\mathbf{x}_4) = 1, y_5 h_1(\mathbf{x}_5) = -1$$

- Set weight: margin large then model weight large, otherwise model weight small: want to make more instances to have larger margins.

- Change data distribution: increase instance weight for small-margin instances.



AdaBoost

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization factor

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

AdaBoost

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \; y_i \; h_t(x_i))$$

instance weights change w.r.t. margin!

where Z_t = normalization factor

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

AdaBoost

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

instance weights change w.r.t. margin!

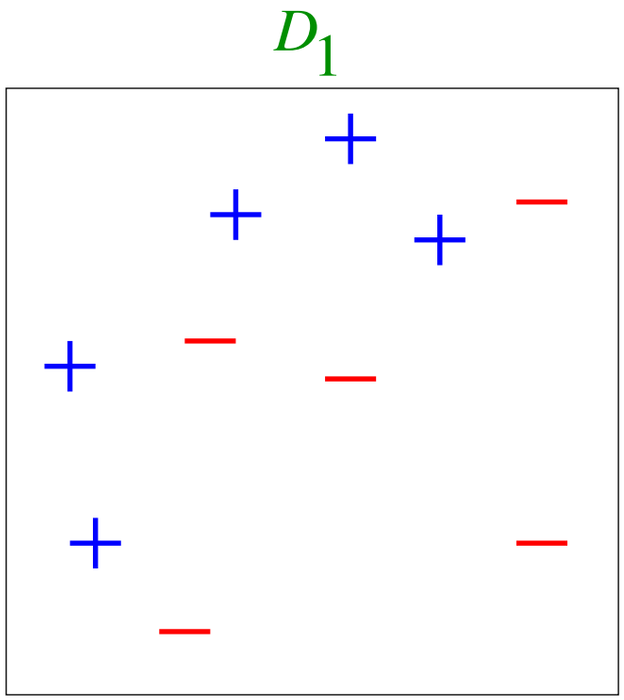
where Z_t = normalization factor

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

model weight set according to error on D_t

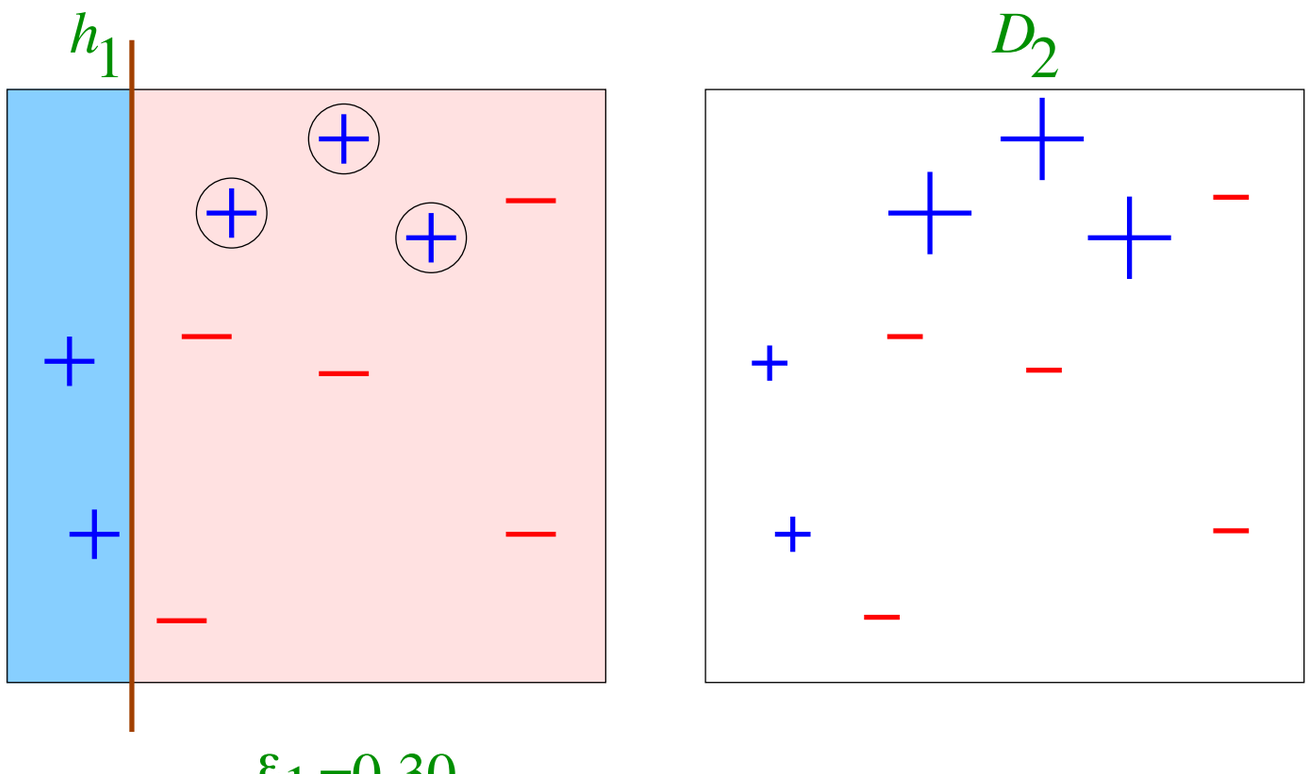
- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Toy Example



weak classifiers = vertical or horizontal half-planes

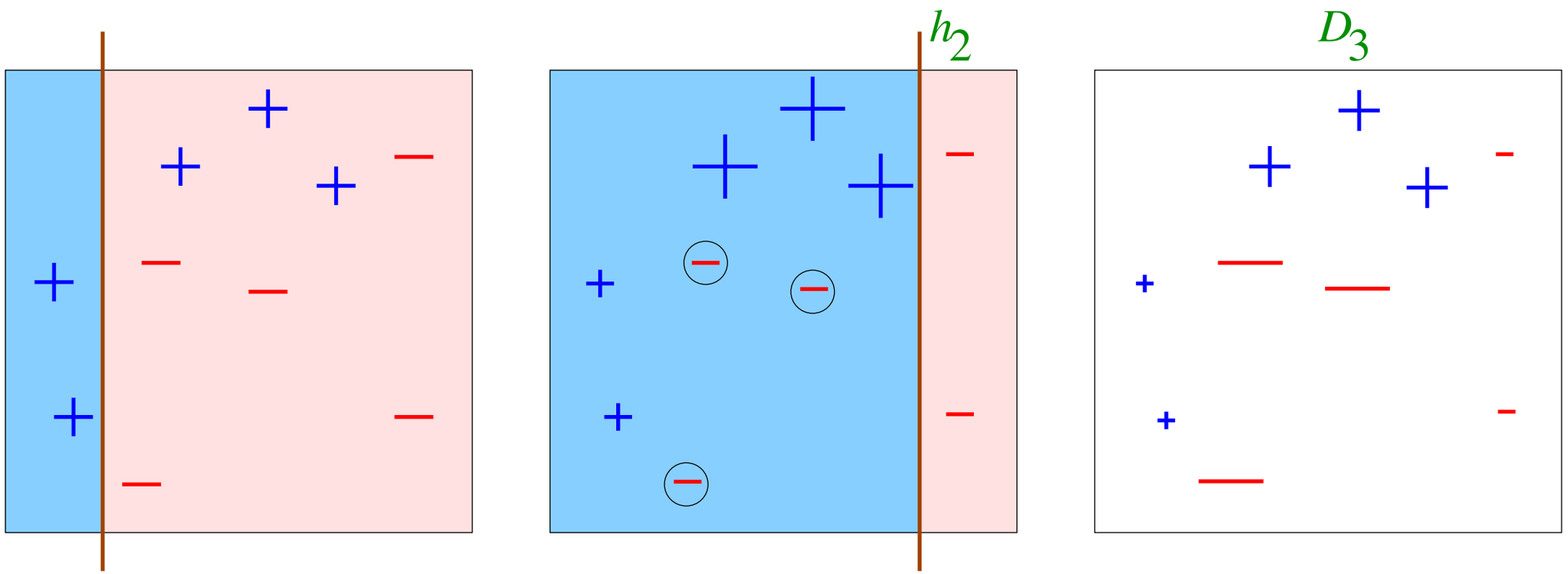
Round 1



$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

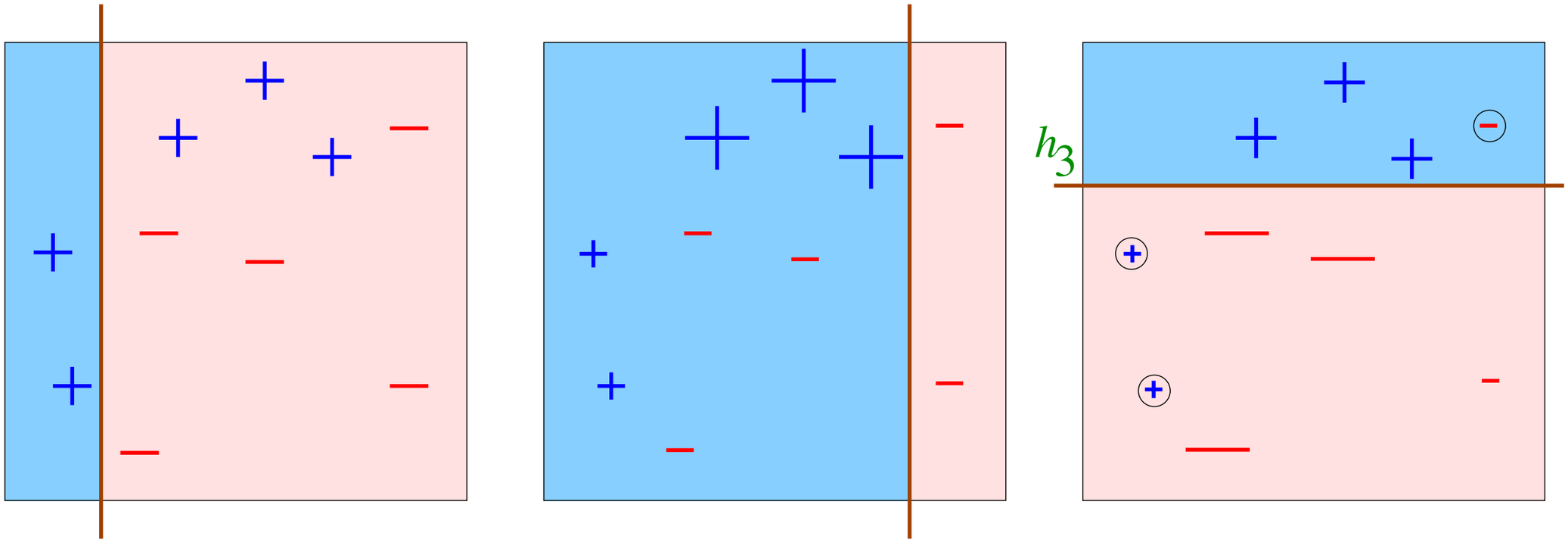
Round 2



$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

Round 3

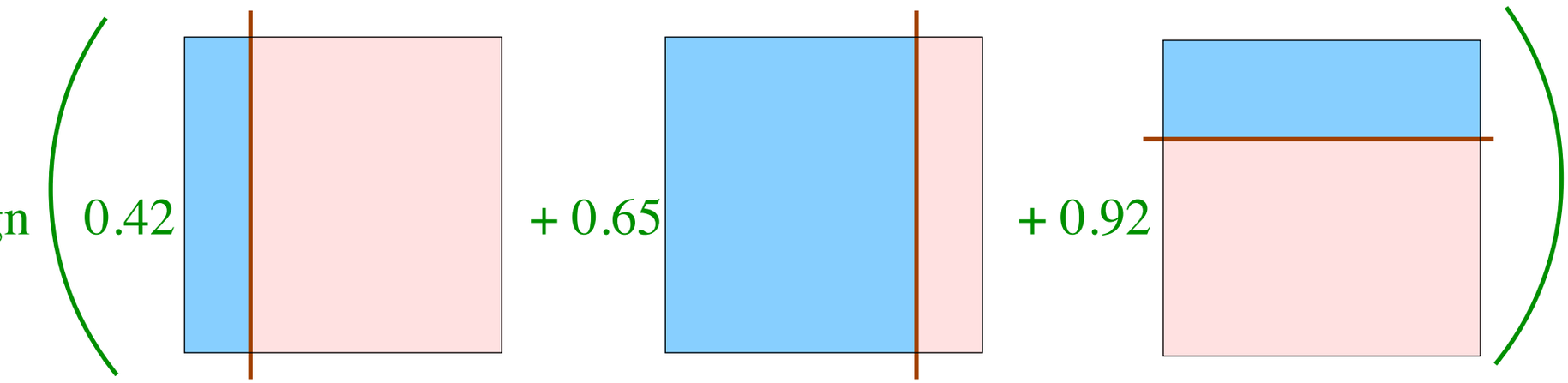


$\varepsilon_3 = 0.14$

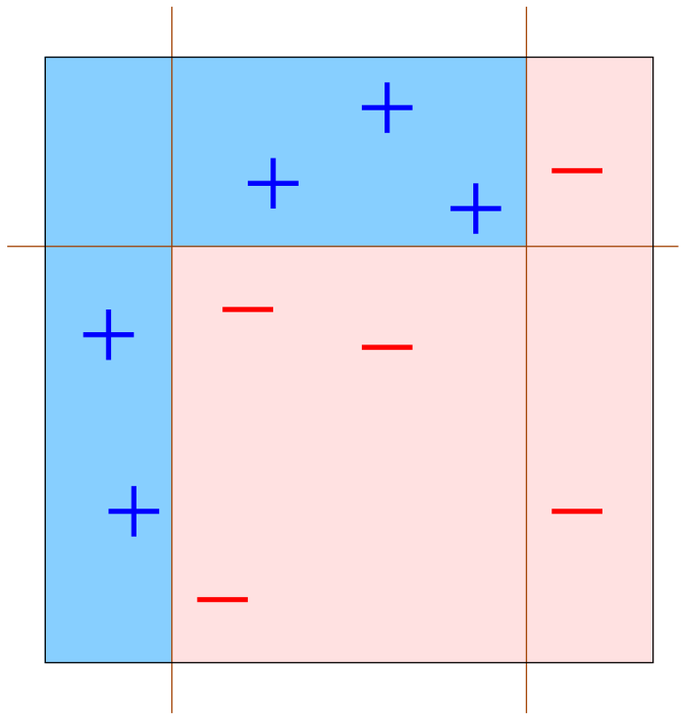
$\alpha_3 = 0.92$

Final Classifier

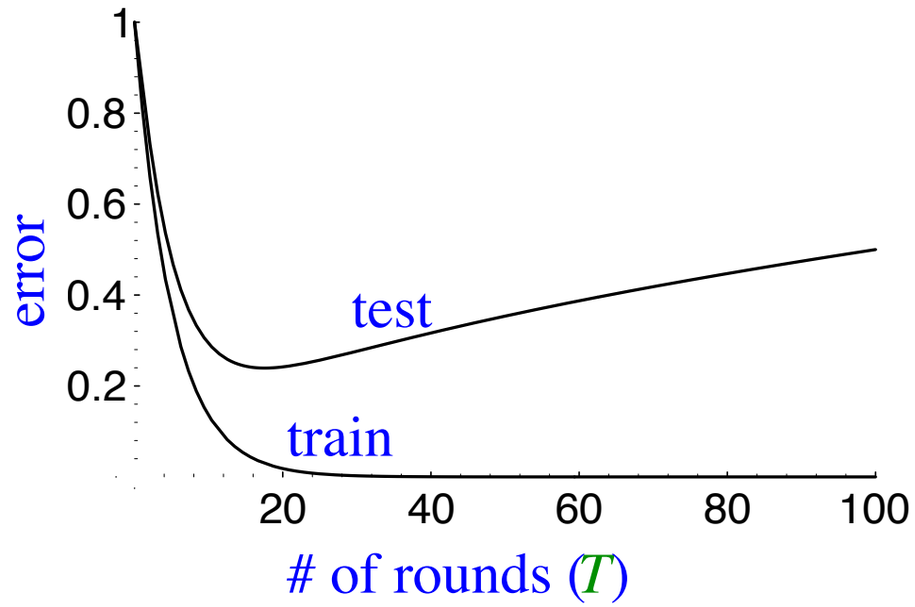
$$H_{\text{final}} = \text{sign}\left(0.42 \; + \; 0.65 \; + \; 0.92 \right)$$



$$=$$



Mystery of AdaBoost

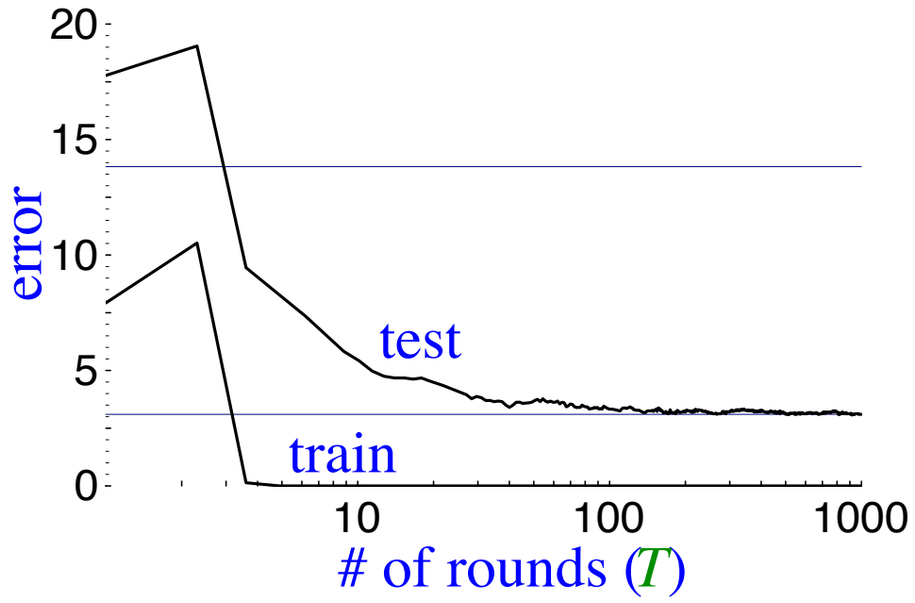


expect:

- training error to continue to drop (or reach zero)
- test error to increase when H_{final} becomes "too complex"
  - "Occam's razor"
  - overfitting
    - hard to know when to stop training

In common machine learning, the testing error could increase when keep training after training error becomes zero: overfitting.

Mystery of AdaBoost



(boosting C4.5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
  - (total size $>$ 2,000,000 nodes)
- test error continues to drop even after training error is zero!

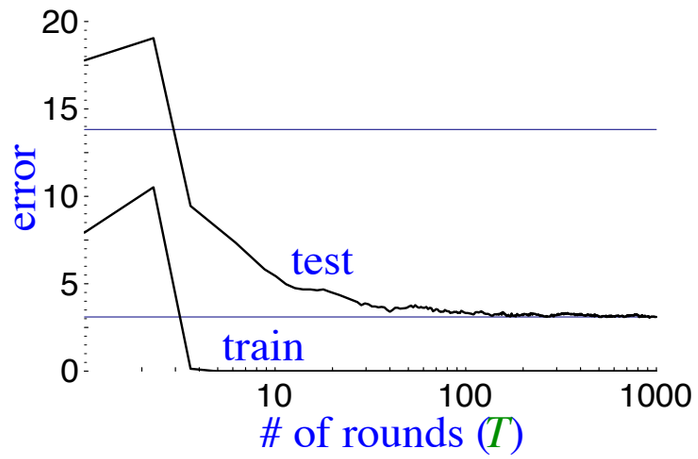
	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

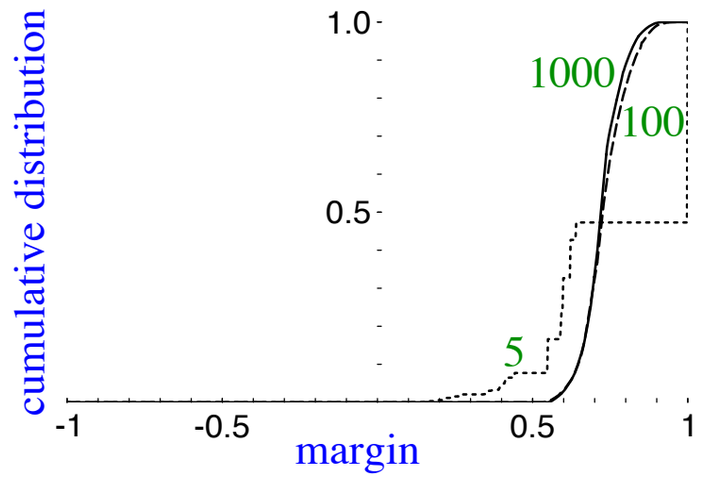
- Occam's razor wrongly predicts "simpler" rule is better

AdaBoost seems to be significantly resistant to overfitting!

AdaBoost Optimizes Margin Distributions

- margin distribution
  = cumulative distribution of margins of training examples



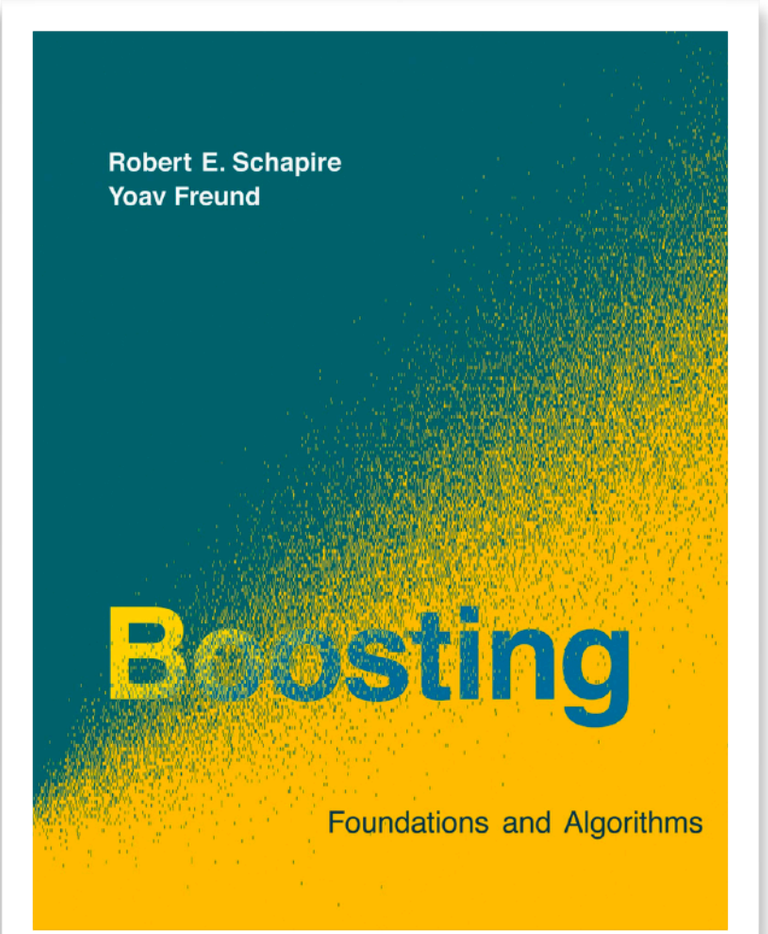


	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

AdaBoost continues to maximizing the margins of the predictions.
This is why AdaBoost is robust to over-fitting!

Many Faces of AdaBoost

- many other ways of understanding AdaBoost:
  - as playing a repeated two-person matrix game
    - weak learning assumption and optimal margin have natural game-theoretic interpretations
    - special case of more general game-playing algorithm
  - as a method for minimizing a particular loss function via numerical techniques, such as coordinate descent
  - using convex analysis in an "information-geometric" framework that includes logistic regression and maximum entropy
  - as a universally consistent statistical method
- can also derive optimal boosting algorithm, and extend to continuous time



Gödel Prize - 2003

Yoav Freund and Robert Shapire

The 2003 Gödel Prize for an outstanding journal article or articles in theoretical computer science is awarded to Yoav Freund and Robert Schapire for their paper
"A Decision Theoretic Generalization of On-Line Learning and an Application to Boosting"
Journal of Computer and System Sciences **55** (1997), pp. 119-139.

Machine Learning: II

- Support vector machine (SVM)
  - Linear separable case
  - Linear inseparable case
- AdaBoost
- Take-home messages

Take-Home Messages

- Margin maximization is a powerful regularization strategy.
- Support vector machine:
  - Large-margin linear classifier.
  - Use strong duality to solve the convex optimization problem.
  - Soft margin & kernel method to obtain non-linear classifiers.
- AdaBoost:
  - learn weak classifiers in sequence to form the voting classifier.
  - Maximizing margin distribution to achieve better generalization.

Thanks for your attention!
Discussions?

Acknowledgement: Many materials in this lecture are taken from
the slides from Dan Klein, Pieter Abbeel, David Sontag and Rob Schapire:
http://ai.berkeley.edu/lecture_slides.html
https://people.csail.mit.edu/dsontag/courses/ml16/slides/lecture3.pdf
https://www.cs.princeton.edu/courses/archive/spring12/cos598A/slides/intro.pdf